National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

NL-339 (r. 88/04)

Canada

The University of Alberta

A COMPUTER MODEL OF KNOWLEDGE ORGANIZATION AND STRATEGY SHIFTS IN NOVICE-EXPERT PROBLEM SOLVING

by

© Peternela Barbara Scharf

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Spring, 1988

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-42862-7

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Peternela Barbara Scharf

TITLE OF THESIS: A Computer Model of Knowledge Organization and Strategy Shifts in Novice-Expert Problem Solving

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1988

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Peternela Scharf

73 Amherst Crescent
St. Albert, Alberta
Canada, T8N 2P7

Date: February 29, 1988

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **A Computer Model of Knowledge Organization and Strategy Shifts in Novice-Expert Problem Solving** submitted by **Peternela Barbara Scharf** in partial fulfillment of the requirements for the degree of **Master of Science.**

Renée E. Elio
Supervisor

Duane Szafron

Alinda Fried

Randy Goebel

Date: Feb. 15, 1988

To Terry

ABSTRACT

Empirical studies which investigate differences between novice and expert problem solvers have suggested that experts not only have more domain knowledge than novices, but that their knowledge is organized in a qualitatively different manner. These qualitatively different knowledge organizations may contribute to the different problem-solving strategies employed by novices and experts in a particular domain. This research presents a process model that simulates both the differences in knowledge organization and the problem-solving strategies that characterize novices and experts in the domain of physics word problems. In particular, the model is based on a framework which naturally and easily accomodates the evolution of expert-like organizations and strategies as a function of problem-solving experience. The model is implemented in a system called EUREKA. EUREKA develops solution methods using means-ends analysis which then become part of a highly interrelated set of problem-type schemas. Initially, superficial features of the problem statement form the basis for both the problem-type schemas and the discriminating features that organize them in the network. As EUREKA solves more problems, the content and the discriminating features change to reflect more fundamental physics principles. This allows EUREKA to shift from a novice-like means-ends strategy to a more expert-like "knowledge development" strategy in which inferences, constraints, and solution methods are triggered by the features of the problem. Thus, EUREKA's framework seems well-suited for capturing the qualitatively different types of knowledge that an expert seems to have. Moreover, the shift to an expert-like strategy emerges as a natural consequence of the shift to a more expert-like knowledge organization. The EUREKA model accounts for several descriptive models of novice-expert differences and suggests a number of empirical predictions.

Acknowledgements

I'd like to acknowledge the invaluable guidance of my supervisor, Dr. R. Elio, and the contributions made by the members of my examining committee, Dr. A. Friedman, Dr. R. Goebel and Dr. D. Szafron. I'd like to also acknowledge the financial support of the Department of Computing Science and the NSERC Operating Grant A0889 to Dr. R. Elio.

Table of Contents

Chapter Page

1. Introduction 1

2. Empirical Studies and Computer Models of Skill Acquisition 5

2.1 Empirical studies on novice and expert differences 5

2.1.1 Task performance differences 6

2.1.2 Problem schema and knowledge organization differences . . 11

2.1.3 Attentional Differences 17

2.1.4 Summary 18

2.2 Models of skill acquisition 19

2.2.1 A general learning framework 21

2.2.2 Tuning patterns 22

2.2.2.1 SAGE 23

2.2.2.2 LEX 25

2.2.2.3 HPM 26

2.2.2.4 Soloway and Riseman's baseball system . . . 27

2.2.2.5 ABLE 29

2.2.3 Learning larger patterns 30

2.2.3.1 STRIPS 30

2.2.3.2 HACKER 32

2.2.3.3 Knowledge Compilation in ACT Theory . . . 34

2.2.3.4 Chunking 39

2.2.4 Learning new organizations of patterns 43

2.2.5 Summary 45

3. A Framework for Knowledge Reorganization 48
- 3.1 Introduction 48
- 3.2 Overview of CYRUS system 50
- 3.3 How E-MOPs are dynamically reorganized 53
  - 3.3.1 Index selection 54
  - 3.3.2 Creating an E-MOP 55
  - 3.3.3 Monitoring the E-MOP network 56
  - 3.3.4 Issues in controlling the mechanisms 56
- 3.4 Summary 57

4. The EUREKA System Design 58
- 4.1 Overview of the model 58
- 4.2 How a problem is represented 59
- 4.3 A sample P-MOP 64
- 4.4 The EUREKA model 67
  - 4.4.1 The semantic network 67
  - 4.4.2 Means-ends analysis 70
  - 4.4.3 Storing the enhanced problem representation . . . 76
  - 4.4.4 Retrieving past problem solving knowledge from P-MOPs . . 83

5. Trace 85

6. Discussion 130

Bibliography 142

List of Figures

Figure Page

2.1 A Sample of a Novice and Expert Protocol 8

2.2 Problems categorized by novices as similar 13

2.3 Problems categorized by experts as similar 14

2.4 Network representation of a novices schema of an inclined plane . 15

2.5 Network representation of an experts schema of an inclined plane . 16

2.6 Three sample strategy transformation heuristics . . . 27

2.7 A generalized plan in a triangle table 32

2.8 SAS Postulate 35

2.9 Declarative Representation of the SAS Postulate . . . 36

2.10 General Productions 37

2.11 A simple three-goal hierarchy for the averaging of two numbers . 41

2.12 Sample chunks created for the hierarchy of Figure 2.11 . . 42

3.1 Sample events 50

3.2 An E-MOP structure 52

4.1 Initial problem representation of Problem1 60

4.2 Enhanced problem representation of Problem1 . . . 63

4.3 Problem1's solution method goal tree 64

4.4 A sample P-MOP 66

4.5 Semantic Network 68

4.6a Control structure for problem solver 72

4.6b Control structure for problem solver 73

4.7 Checking the information on a P-MOP against an incoming problem representation 79

4.8 Arriving at a stored problem and creating a new P-MOP . . 81

Figure Page

4.9 Collapsing a P-MOP 82

5.1 P-MOP network after Problem1 is indexed 88

5.2 P-MOP network after Problem2 is indexed 91

5.3 P-MOP network after Problem3 is indexed 94

5.4 P-MOP network after Problem4 is indexed 99

5.5 P-MOP network after Problem5 is indexed 104

5.6 P-MOP network after Problem6 is indexed 108

5.7a P-MOP network after Problem7 is indexed 112

5.7b P-MOP network after Problem7 is indexed 113

5.8a P-MOP network after Problem8 is indexed 115

5.8b P-MOP network after Problem8 is indexed 116

5.9a P-MOP network after Problem9 is indexed 119

5.9b P-MOP network after Problem9 is indexed 120

5.10a P-MOP network after Problem10 is indexed 123

5.10b P-MOP network after Problem10 is indexed 124

5.11 P-MOP network 127

5.12 Network representation of an experts schema of an inclined plane . 128

Chapter 1

Introduction

Within the last ten years, artificial intelligence research has been concerned with the representation and use of domain-specific knowledge for the solution of complex problems. This trend was heralded by early research in complex knowledge-based systems in domains such as chemistry [Buchanan & Feigenbaum, 1978] and medicine [Davis, Buchanan & Shortliffe, 1977]. Since then, much attention has been given to fundamental issues regarding how large amounts of domain-specific knowledge can be organized and how that organization can support the efficient recognition and application of relevant knowledge to the problem at hand. The last decade of empirical research on human problem solving has shown a similar interest in complex, knowledge-rich domains. Since the seminal work on chess expertise by de Groot [1966] and Simon and Chase [1973], the general paradigm of investigating and characterizing the differences between novice and expert problem solvers has been extended to areas such as mathematical problem solving [Anderson, Greeno, Kline & Neves, 1981; Schoenfeld & Herrmann, 1982], physics problem solving [Simon & Simon, 1978; Larkin, 1979; Larkin, McDermott, Simon & Simon, 1980; Chi, Feltovich & Glaser, 1981; Chi, Glaser, & Rees, 1982; Larkin, 1983], programming [McKeithen, Reitman, Rueter & Hirtle, 1981; Anderson, Farrell & Sauers, 1984], medical diagnosis [Pople, 1977; Johnson et al., 1981], and the social sciences [Voss, Greene, Post & Penner, 1983]. Both the trend in knowledge-based problem solving in artificial intelligence as well as the early work on chess expertise supports the generally accepted view that expert problem solving rests largely on having appropriate domain-specific knowledge and not on any unusual intellectual abilities. However, recent empirical work on novice-expert differences has indicated that an expert not only knows more than a novice, but that there is a qualitative difference in how experts and novices organize their knowledge about some domain and use that knowledge during problem solving.

The goal of this dissertation is to provide a psychologically-plausible process model that accounts for, at a qualitative level, both the differences in knowledge organization and the problem-solving strategies that characterize novices and experts in the domain of physics problems. The model is based on a framework which naturally and easily accomodates the evolution of expert-like organizations and problem-solving strategies as a function of problem-solving experience. The model is implemented in a system called EUREKA (Expertise Using REstructuring of Knowledge). The characterizations of novice and expert physics problem solving are based on certain empirical findings and descriptive psychological models that will be reviewed later in the thesis. This work makes a fundamental assumption that an important aspect of expertise is the restructuring of existing knowledge. Thus, this dissertation is concerned with exploring a framework of learning mechanisms that can account for changes in the organization of knowledge of novices and experts.

EUREKA begins with a set of unorganized physics concepts, equations, and inferences. This knowledge represents the initial "textbook" knowledge a novice is likely to have about force and energy problems. EUREKA uses means-ends analysis to solve physics problems using this knowledge. Each problem-solving experience is stored in a complex discrimination network according to its common and discriminating features with previous problem-solving experiences. The network is restructured to accommodate and integrate each new experience. New commonalities and discriminating features are abstracted during this integration process. In this manner, EUREKA learns a highly-interrelated set of problem-type schemas containing associated solution methodologies. Initially, superficial features of the problem statement form the basis for both the problem-type schemas and the discriminating features that organize them. An examination of EUREKA's evolving network indicates that the content and the organization of schemas change to reflect more fundamental physics principles as EUREKA solves more problems. This evolving organization of problem types and discriminating features allows EUREKA to shift from an unguided means-ends strategy to one in which inferences and methodologies relevant to the solution are triggered by the features of the problem.

Some of the fundamental ideas in EUREKA's approach are similar to Novak's views of expert problem solving in designing the ISAAC system [1977, 1980]. Novak attributes the superior performance of experts to their ability to recognize that a given problem is an instance of a class of decomposable problems. In particular, this strategy can be viewed as an attempt to apply the appropriate decomposition so that the relevant physics principles can be identified. This type of approach is more efficient than the means-ends analysis approach of a novice because correct equations are more likely to be identified. Novak [1980] proposed that a discrimination net might support the recognition of what the appropriate decomposition is for a particular problem.

The EUREKA system is important from a cognitive science viewpoint because it can account for empirical results on novice to expert shifts in both knowledge organization as well as problem-solving strategies. Researchers have offered descriptive models of novice and expert knowledge structures, but do not offer processes by which these knowledge structures evolve [e.g., Chi et al., 1981]. Other researchers have provided computer models which shift from novice-like strategies to expert-like strategies, but these models do not address the broader issue of restructuring knowledge around evolving problem prototypes or schemas [e.g., Larkin, 1981; Anderson et al., 1981]. From an artificial intelligence perspective, this work explores mechanisms by which a skill can be acquired through problem-solving experiences. I have taken a scheme for dynamically reorganizing episodic memory and extended it as a learning framework for skill acquisition in problem solving. In this respect, EUREKA is a kind of "learning by doing" system. However, few of the learning frameworks and skill acquisition systems in the AI literature encompass both a shift in problem-solving strategies and a qualitative restructuring of knowledge around multiple problem prototypes in a given domain.

This work also has a practical significance. The ability of a system to develop, refine, and reorganize problem prototypes and associated methodologies is important for designing knowledge-based systems that learn from their own problem-solving experiences or from observing a human expert. EUREKA could be viewed as a knowledge-based problem solver.

This dissertation is organized as follows. The next chapter reviews two areas of research central to this work: empirical studies on novice-expert differences in physics problem solving, and computer models of skill acquisition. The particular empirical results in physics problem solving that EUREKA aims to simulate are summarized in Chapter 3. This is followed by a detailed presentation of the framework for knowledge reorganization on which EUREKA is based. Chapter 4 describes EUREKA's design. Chapter 5 provides a trace of EUREKA's performance on 10 problems, which demonstrates its evolution from a novice-like knowledge organization and problem-solving strategy to a more expert-like knowledge organization and problem-solving strategy. The concluding chapter evaluates EUREKA 's design and assumptions as a viable process model, relates EUREKA to other skill acquisition systems, discusses some empirical predictions made by EUREKA, and summarizes the important contributions of this work.

Chapter 2

Empirical Studies and Computer Models of Skill Acquisition

This literature review surveys two areas of research. Because the goal of this work is to develop a framework for modeling some specific empirical results in the physics domain, it is important to review the empirical literature on differences between novices and experts. The important empirical findings about novice-expert differences are that experts and novices have (a) qualitatively different problem-solving strategies and (b) qualitatively different organizations of knowledge.

The second area I review concerns computer models of skill acquisition. Models and theories which are covered in this chapter propose various mechanisms by which performance on a particular task improves with experience. However, it will become apparent that many of these mechanisms are grounded in rather similar underlying ideas - detecting similarities and differences in patterns, generalizing and specializing patterns, forming large patterns from small patterns, and caching or saving methods. Some of the models directly address the dynamic reorganization of knowledge as a function of experience and accomplish this reorganization through the recognition of similarities and differences. EUREKA is based on this type of model.

2.1 Empirical studies on novice and expert differences

de Groot's [1966] study of novice and expert chess players is, perhaps, the seminal work in this area. de Groot found that experts remember larger patterns of chess pieces than novices. Simon and Chase [1973] extended this research with their findings that experts have a better memory for chess patterns *only if* the pattern is meaningful. In their experiments, experts did not remember random placements of chess pieces any better than novices. These general findings have been replicated by studies of expertise in other domains. For example, a study conducted by McKeithen, Reitman, Rueter, and Hirtle [1981] found that experts remember meaningful computer programs better than do novices.

On the basis of these findings, cognitive psychologists have proposed that experts *do not* have any exceptional general cognitive abilities, for instance, better memory. However, experts *do* have large amounts of task specific knowledge organized in meaningful chunks. Simon and Chase [1973] estimate that a chess master can recognize at least 50,000 chess patterns! This is like having 50,000 if-then rules available to match different configurations (or patterns) of chess pieces. From a processing viewpoint, this would seem to present a "paradox of interference" - with so much knowledge, experts should be painstakingly slow at solving problems, when in fact they are not [Smith, Adams & Schoor, 1978; Reder & Anderson, 1980]. This suggests that knowledge organization and retrieval are key components in understanding the mechanisms of skill acquisition and expertise. The empirical studies reviewed in the next sections support the notion that experts not only know more but have qualitatively different problem-solving strategies and knowledge organizational schemes. These empirical studies investigate three areas of expert-novice differences: task performance differences, knowledge organization differences, and attentional differences.

2.1.1 Task performance differences

There are two very basic and intuitive differences in task performance between experts and novices. First, experts solve problems faster than novices do [Simon and Simon, 1978], and second, they commit fewer errors than novices [Chi, Glaser and Rees, 1982]. The studies that follow highlight two other important task performance differences on how experts and novices solve physics problems. Experts appear to reason forward from what is known in a problem whereas novices focus on what is desired in the problem, select an equation to reduce this difference, and continue on in this means-ends fashion.

Experts also seem to engage in a qualitative analysis of a problem prior to using equations, whereas novices appear to engage immediately in equation retrieval and manipulation.

Simon and Simon [1978] and Larkin, McDermott, Simon, & Simon [1980] found qualitatively different problem-solving strategies for experts and novices. In both studies, experts used a strategy which the researchers characterize as a "knowledge development" strategy. Using the knowledge development strategy, a problem solver works forward from the known variables of the problem statement, generating equations from which additional quantities become known. The expert continues to work in this manner until the original desired quantity of the problem has been derived. In contrast, novices tend to use a means-ends approach, starting with equations that solve for the desired quantity of the problem. If the selected equation contains an unknown variable, then another equation is selected to solve for this variable. The process continues until an equation is selected in which all variables are known. An example illustrates the two different methods. Suppose that a problem asks for the value of E, and the known variables are B,C, and D. The following two equations are available:

1) $A = B + C$

2) $E = D * A$

A knowledge development strategy selects equation 1 because B and C are known. As a result variable A is derived. Equation 2 is then selected because D and A are known. This results in the derivation of the desired variable E. In contrast, a novice uses a means-ends strategy. The goal of this problem is to solve for E, the desired variable. An operator is required to reduce this difference. Therefore, equation 2 is selected. In equation 2 D is known and A is unknown. An operator is required to reduce the difference of A being unknown. This results in the selection of equation 1. Because both B and C are known in equation 1, A is derived. The variable A is then substituted back into equation 2 and the original desired variable E is derived.

Larkin [1979] collected protocols of two experts and one novice while they were solving mechanics problems. She found that experts apply a qualitative analysis to a problem prior to working with equations. Qualitative analysis of a problem includes planning, drawing inferences to generate information not explicitly stated in the problem, and checking the solution.

Expert and novice protocols for the solution of a problem are shown in Figure 2.1. Notice that the novice immediately starts generating equations and instantiating variables. In contrast, the expert's protocol does not mention equations, but rather components of the problem. Apparently, the expert seems to imagine what will happen in terms of the forces acting in the problem. The expert generates additional information such as "the total force must be zero" and "this horizontal component must be larger than some minimum amount set by the friction coefficient."

Problem Statement: The handle of a floor mop of mass m makes an angle ø with the vertical direction. Let μ_s and μ_k be the coefficients of static and kinetic friction between the mop and the floor. Find the magnitude of the force F directed along the handle that is required to slide the mop across the floor with constant velocity.

Novice Solution (beginning): So I have a frictional force. And since this is moving, I know that the frictional force is equal to μ_k times the normal force. And the normal force is merely the mass of the mop, so I can say that μ_k times mg is equal to the frictional force.

Expert Solution (beginning): Then in fact the total force on the thing must be zero. Looking at various cases, if the force is vertical, then I don't care how hard you push it, it will never go any place. But if you make the angle ø larger, you get more of a horizontal component. You essentially have a chance to make it move horizontally. But this horizontal component must be larger than some minimum amount set by the friction coefficient.

Figure 2.1
A Sample of a Novice and Expert Protocol
(From Larkin, 1979, P285-6)

In contrast to Larkin's findings, a protocol analysis study by Chi and her associates [1982] found that novices also spend time analyzing a problem qualitatively. However, novices often either fail to generate a necessary inference or generate faulty inferences. Voss, Greene, Post, and Penner [1983] also found that experts spend considerable time developing a problem representation before suggesting solutions. In contrast, novices immediately start suggesting solutions.

Detailed studies of experts' problem-solving protocols reveal that experts often use a "scientific" problem representation when solving a problem [Larkin, 1983]. According to Larkin, a scientific representation contains abstract physics concepts such as forces and energies. She found that experts, when confronted with a difficult problem, construct such representations prior to using equations.

Based in part on these findings, Larkin [1985] proposed that there are four stages of representation during problem solution. A *verbal* representation is the first stage and is a literal representation of the problem statement. A *naive* representation characterizes the second stage. This usually takes the form of a sketch that depicts the objects in the problem, such as blocks, planes, pulleys and springs, and the relationships between the objects. The *scientific* representation of the third stage contains abstract physics concepts such as forces and energies that are inferred by the problem solver and can be depicted in a free-body diagram. The *mathematical* stage is the final representation of the problem. In this stage, equations relate the variables of the problem to produce the final answer.

In going further with this framework, Larkin [1985] proposes that a main difference in problem-solving performance between experts and novices is due to the inability of a novice to construct and use a scientific representation. A scientific representation, in her view, provides a mental model with powerful attributes. It describes the important features of a problem and can provide a basis for generating equations. Larkin believes that a problem-solver using a scientific representation, accesses equations that relate the objects, features, and relations of the naive representation to entities of the scientific representation. For example, the following equation:

Potential Energy = mass * gravity * height

states that potential energy, an entity of a scientific representation, is a function of mass, gravity, and height, entities of the naive representation. The knowledge development strategy of an expert enables the transformation of a naive representation to a scientific representation. Larkin also believes that when novices go directly from a naive representation to a mathematical representation, the only cue as to what would be a useful equation is whether the equation contains the desired quantity of the problem. Means-ends provides the strategy in which to do this. Thus, stages of representation can be used to explain differences in the problem-solving strategy of novices and experts.

To summarize, two important results emerge from the task performance studies described above. Experts seem to use a knowledge development strategy when solving a problem whereas novices appear to use means-ends analysis. What "develops" in knowledge development can be viewed as a richer and qualitatively different representation of the problem. Secondly, experts appear to engage in qualitative analysis prior to using equations. They try to fit a problem into a scientific representation that focuses on abstract physics concepts, such as potential energy, which then assists in the generation of equations for the solution of the problem. In contrast, a novice seems to move directly from a naive representation, centered around physics attributes such as mass, to a mathematical representation of the problem. This is characterized by the novices strategy of generating candidate equations. Although many novice-expert differences in problem-solving strategies can be attributed simply to the expert knowing more, problem-solving strategies are influenced by qualitatively different organizations of the same knowledge. I turn now to studies that investigate differences in knowledge organization between experts and novices.

2.1.2 Problem schema and knowledge organization differences

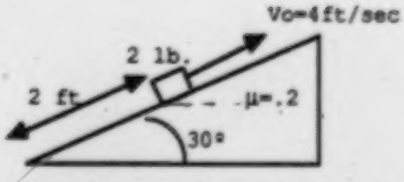
Several studies suggest that experience with solving problems in a given domain leads to the evolution of problem schemas. In general terms, a schema can be thought of as an organized unit of knowledge and as such, relates directly to similar concepts like frames [Minsky, 1975] or scripts [Schank, 1977] in artificial intelligence. A more detailed discussion of schemas appears in Section 2.2.1. Studies done by Hinsley, Hayes, and Simon [1978] indicate that students classify algebra problems into *types* and can perform this classification after reading only part of the problem statement. Expert medical diagnosticians also seem to have prototypes of particular diseases or disease combinations. These categories are triggered by patient symptoms and are used to direct hypothesis generation during diagnosis [Pople, 1977; Johnson et al., 1981]. This has a similar flavor to the knowledge development notions provided by Larkin et al. [1980].

The notion of schemas is a recurrent theme in the studies that follow. These studies investigate differences in novice and expert physics problem schemas. The important finding is that experts categorize problems as similar if the problems are solved by the same physics principle, whereas novices group problems that have the same surface features. Experts also tend to have more well-specified solution methodologies that are organized and accessed by these physics principles.

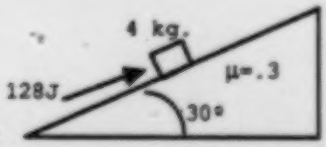
Larkin [1979] measured the time intervals between equation generation for experts and novices as they solved mechanics problems. She found that experts generated many pairs of equations with short time intervals, suggesting that these equations may be stored together. The time interval between pairs of equations generated by the novices varied randomly. These findings support the idea that, generally, an expert's knowledge organization is different from that of a novice. More precisely, notions such as schemas or "associated chunks" are proposed to account for these reaction time differences. Larkin's findings and this type of interpretation are consistent with other reaction time studies of expertise [de Groot, 1966; Simon and Chase, 1973].

Chi et al. [1981] conducted studies with the aim of discovering the ways in which problem-type schemas, if they existed, differ for novices and experts. Like Hinsley et al. [1978], they conducted categorization experiments to determine how novices and experts classify problems. Novices and experts were asked to sort problems into categories based on the similarity of solution and to give reasons for the categorization of the problems. In this study, novices categorized problems according to what Chi et al. call the "surface features" of the problem. Surface features include objects, relationships between objects, and physics terms mentioned in the problem. Figure 2.2 shows an example of two problems that novices regard as similar. Typical category labels that novices often used to classify problems include "springs", "inclined planes", "velocity and acceleration", "friction", and "angular motion".

In contrast, experts do not use the surface structure of the problem for classification. As shown in Figure 2.3, the physics problems included in an expert category generally have no similarity in the problem statement or surface characteristics of the problem. Problems are classified according to what Chi et al. call their "deep structure", the major physics principle that would be used to solve the problem. Examples of expert category labels include "Newton's second law", "energy principles", and "momentum principles". Schoenfeld and Herrmann [1982] found the same type of category differences in the domain of mathematical problem solving.



Problem 7(23) A 2.0 lb block is given an initial speed of 4.0 ft/sec up an inclined plane starting from a point 2.0 ft from the bottom as measured along the plane. If the plane makes an angle of 30° with the horizontal and the coefficient of friction is 0.20, how far up the plane will the block travel?



Problem 7(35) A 4.0 kg block starts up an 30° incline with 128 joules of kinetic energy. How far will it slide up the plane if the coefficient of friction is 0.30?

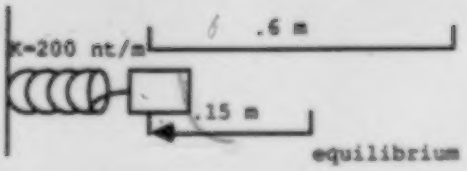
Novices' Explanations for their Similarity Groupings

Novice 1: "These deal with blocks on an incline plane"

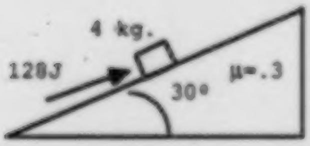
Novice 5: "Inclined plane problems, coefficient of friction

Novice 6: "Blocks on inclined planes with angles"

Figure 2.2
Problems categorized by novices as similar
(Adapted from Chi et al., 1981)



Problem 6(21) A 2.0 kg block is forced against a horizontal spring of negligible mass, compressing the spring by 15 cm. When the block is released, it moves 60 cm across a horizontal tabletop before coming to rest. The force constant of the spring is 200 nt/meter. What is the coefficient of sliding friction between the block and the table?



Problem 7(35) A 4.0 kg block starts up an 30° incline with 128 joules of kinetic energy. How far will it slide up the plane if the coefficient of friction is 0.30?

Experts' Explanations for their Similarity Groupings

Expert 2: "Conservation of Energy"

Expert 3: "Work-Energy Theorem They are all straight-forward problems."

Expert 4: "These can be done from energy considerations. Either you should know the Principle of Conservation of Energy, or work is lost somewhere."

Figure 2.3
Problems categorized by experts as similar
(Adapted from Chi et al., 1981)

Novices and experts were presented with a category label and asked to tell everything they knew about the subject [Chi et al., 1981]. Figures 2.4 and 2.5 present an interpretation of a novice and an experts "dump" of inclined plane knowledge[1]. Figure 2.4, shows that a novice knows a great deal about the category inclined planes. The novice has variables to instantiate, angle of incline, the length of the plane, the mass and height of a block on the plane, and the coefficient of friction of the plane. The novice also mentioned normal force and a force due to a pulley as possible forces acting on a block. A physics principle, Conservation of Energy, was mentioned as a description of a problem.

---

[1] The "meaning" of the arrow heads is not clear from the Chi et al. article, but it may reflect the order of output in the protocol and Chi et al.'s interpretation of this ordering as associative paths.

This Figure has been removed because of the unavailability of copyright permission.

Figure 2.4
Network representation of a novices schema of an inclined plane. (From Chi et al., 1982, P57)

Like the novice's knowledge, the expert's knowledge base also contains surface information about inclined planes. In contrast with the novice, the knowledge of the expert contains physics principles, their conditions of applicability, and procedural knowledge associated with the physics principle (Figure 2.5). Experts' protocols explicitly mention such conditions and methods. For example, the expert's knowledge base suggests that an inclined plane problem can be solved using Newton's force laws. The expert knows that if acceleration is involved, then Newton's second law is relevant, and the forces acting on the body can be equated to the body's mass and acceleration.

This Figure has been removed because of the unavailability of copyright permission.

Figure 2.5
Network representation of an experts schema of an inclined plane. (From Chi et al., 1982, P58)

Chi et al. [1981] also analyzed the knowledge dump by converting the novices' and experts' statements into if-then rules. This type of analysis highlights the kinds of actions triggered by features of the problem. One type of action that might result can be called procedural or "how to" knowledge. From this analysis, it was evident that novices lack procedural knowledge. Novices' rules did *not* contain solution procedures as their actions but rather actions to find specific unknowns, such as the mass of the block or the angle of the incline. An example of a novice production is:

> If the problem involves an inclined plane
> Then find angle of incline with horizontal.

In contrast, experts' rule actions contained explicit solution methods, such as sum all the forces acting on the body. An example of an expert production is:

> If the problem involves an inclined plane
> Then a) expect something rolling or sliding up or down;
> b) use F=MA;
> c) use Newton's 3rd Law.

These studies suggest that novices and experts have qualitatively different schemas organized in qualitatively different ways. Novice schemas lack complete knowledge of physics principles, their conditions of applicability, and solution procedures. This is precisely the knowledge that seems to characterize experts' problem solving schema.

2.1.3 Attentional differences

The studies outlined in the previous sections suggest that experts and novices perform differently when solving a problem and the knowledge base of novices and experts differ in both content and organization. The studies in this section attempt to discover if novices and experts attend to different features of the problem. These studies were conducted with the goal of determining the perceived important features of problems and keywords in the problem statement.

Chi et al. [1981] asked novices and experts to read a problem, give a "basic approach" to the solution of the problem, and indicate the features of the problem that lead to their basic approach. They found that novices and experts mention very different features as important to the solution of the problem. Novices mention the objects and physics terms found directly in the problem statement, such as "inclined plane", "pulley", and "friction". Experts tended to mention states and conditions described by the problem, such as "no initial conditions" or "given final conditions". Some of the features mentioned by experts are not explicitly stated in the problem. Chi et al. call these derived features "second-order features". Examples of second-order features are, "before and after situation" and "no external forces". Novices and experts also differed on their basic approach. Basic approaches of novices were either global statements, such as "First, I figured out what was happening...", or detailed sets of equations that they would use to solve the problem [Chi, Feltovich and Glaser, 1981, P142]. In contrast, an expert's basic approach was the identification of a major physics principle such as a force or energy principle.

In another study, novices and experts were asked to circle the important words in a problem statement that would help them judge the difficulty of a problem. Unlike the previous study, which showed no overlap in the features mentioned by novices and experts, keywords circled by experts were a subset of the keywords novices circled. These studies indicate that novices may be considerably less discriminating then experts in identifying the important keywords of the problem statement.

2.1.4 Summary

The empirical studies discussed in the preceding sections highlight many differences between experts and novices. Novices and experts use different strategies to solve a problem. Chi et al's. [1981] analysis suggest that novices also lack the elaborate declarative knowledge and procedural knowledge of an expert. Although both novices and experts organize their knowledge into categories, novice categories seem to be organized around surface features whereas expert categories seem to be organized around physics principles and other implied aspects of the problem.

Chi, Feltovich and Glaser [1981] believe that the way a person's knowledge is organized has an important impact on their problem-solving ability. They state "A *problem representation* is a cognitive structure corresponding to a problem, constructed by a solver on the basis of his domain-related knowledge and its organization" [Chi, Feltovich & Glaser, 1981 p.121-122]. Chi et al. believe that, during problem-solving, a problem is categorized by a preliminary analysis of the problem features. Once a preliminary category is activated, the declarative and procedural knowledge in this schema assists in the solution of the problem. According to this view, then, the problem representation constructed by the problem-solver is influenced by the knowledge available to the problem-solver and the organization of this knowledge. The knowledge contained in a schema and the organization of schemas differ between novices and experts. "Thus, expert-novice differences may be related to poorly formed, qualitatively different, or nonexistent categories in the novice representation. ... much of expert power lies in the expert's ability to quickly establish correspondence between externally presented events and internal models for these events." [Chi, Feltovich and Glaser, 1981, p 122-123].

2.2 Models of skill acquisition

The empirical research presented in the previous section characterizes expert-novice differences in the framework of some general cognitive structures. Not only do novices and experts seem to have different schemas, but their organization of these schemas also varies. For this dissertation, I take these empirical findings and interpretations as a description of the "endpoints" on a continuum of the skill involving the solution of physics problems. What is important, from a learning viewpoint, is a process model that could plausibly serve as the transition mechanism between the novice endpoint and the expert endpoint.

The acquisition of expertise can viewed as a process of skill acquisition, the improvement of performance with experience. Skill acquisition can be regarded as a form of inductive learning, because every new experience is, in a way, an example. This is especially evident when skill acquisition involves the "abstraction" of a new concept that did not exist prior to a set of experiences. Thus, it is not surprising that two very general learning mechanisms - generalization and discrimination - are employed in most skill acquisition systems. Typically, these mechanisms are syntactic procedures operating on if-then patterns in order to improve performance. Generalization takes an overly-specific pattern and generalizes it so that it applies in more situations. In contrast, discrimination changes an overly-general pattern into a more specific one by restricting the situations in which it can apply. Discrimination requires that trace information be stored in order to compare a good application of a pattern to a bad application. The exact mechanisms by which generalization and discrimination occur differ from system to system, but they exist, functionally, in some form or another.

There are aspects of skill acquisition that distinguish it from inductive learning. For example, people get *faster* with practice. Newell and Rosenbloom [1981] have shown that this phenomenon is modeled by a power law across a variety of cognitive and motor skills. Thus, this kind of "automaticity" is a very real aspect of skill acquisition. Process models that address the automaticity associated with human skill acquisition do so by positing mechanisms that qualitatively change the representation and organization of knowledge. These qualitative changes in knowledge representation and organization seem necessary to account for the effect that increases in knowledge does not slow down problem solving.

In order to present the skill acquisition systems, I will group them into three general categories:

1) Tuning Patterns - the conditions that specify when to use a pattern (e.g., a problem-solving operator) are tuned such that the pattern achieves better performance.

2) Learning Larger Patterns - forming a large pattern of actions or operators from a series of actions or operators that have been applied.

3) Learning New Organizations of Patterns - commonalities are abstracted and knowledge is organized by differences from these commonalities.

The boundaries of these categories are, of course, fuzzy and the systems discussed below may well fall into more than one category. Before presenting the individual models of skill acquisition as represented by the above categories, I will present a general theoretical framework in which learning can be viewed. This framework will be useful in analyzing the skill acquisition systems that follow.

2.2.1 A general learning framework

Rumelhart and Norman [1978] provide a theoretical framework that defines learning in terms of changes that occur in the knowledge base. In their framework, knowledge is a network of interrelated schemas. Schemas can be thought of as packages of related knowledge. A schema is an active knowledge structure that defines a relation among its parts which consist of variables, constants, and other schemas. A schema's slots (variables) are filled by incoming information. Slots can have constraints associated with them that define the range of their acceptable values and default values to be used when values for the slot cannot be determined. Schemas that comprise the knowledge base are active knowledge structures that assist in the processing and comprehension of knowledge. They guide the selection of an appropriate schema for a situation by responding to data that is relevant. Once an appropriate schema is selected, it guides the processing of the situation. This view of schemas as actively directing understanding or processing is similar to Minsky's [1975] frame theory and subsequent frame-based theories, such as Schank's scripts [1977].

Rumelhart and Norman identify three types of learning: accretion, tuning, and restructuring. Accretion deals with the accumulation of knowledge, encoding new information based on existing information contained in the knowledge base. Accretion occurs when existing schemas are adequate for the interpretation of the new information. These existing schemas guide the comprehension and storage of new information.

Tuning changes existing knowledge, encoded within a schema, in order to better cope with the functional demands made on the schema. Tuning involves a generalization or a discrimination process. Generalization extends the range of the constraints of a variable or changes a constant to a variable. Discrimination is a specialization process that reduces the range of the constraints of a variable or changes a variable to a constant. Thus, tuning changes the content of a schema but not the relational structure of the schema within the knowledge base.

Restructuring modifies the organization of knowledge structures. Restructuring occurs when new information does not fit existing schemas or when a group of schemas needs reorganization. In the first case, a process called *patterned generation* is used to create a new schema. Under this process, the existing schema that best fits the situation is used as a pattern for the new schema. Modifications are made to this schema to accommodate the new information. In the second case, a process called *schema induction* restructures existing schemas. Schema induction creates a new schema in response to noticing regularities or co-occurances between existing schemas. Because of the difficulties involved in discovering regularities, Rumelhart and Norman believe that most new schema are created through patterned generation rather than schema induction.

Rumelhart and Norman provide a general theoretical framework within which learning mechanisms can be viewed, but they do not provide a process model or specific mechanisms. The skill acquisition systems that follow improve their performance by using learning mechanisms that tune or restructure knowledge as a function of experience.

2.2.2 Tuning patterns

The five systems discussed in this section tune knowledge in order to achieve better performance. SAGE [Langley, 1983, 1987] uses a discrimination mechanism to tune its problem-solving operators. LEX [Mitchell, Utgoff, & Banerji, 1983] makes use of both discrimination and generalization mechanisms to acquire and refine problem-solving operators. Both of these systems tune the conditions of applicability of an operator in order to achieve better performance. HPM [Neches, 1987] uses general heuristics in order to modify operators into a more efficient form. These heuristics tune both the conditions and actions of operators. Soloway and Riseman [1977] developed a baseball system that uses general schemas to guide the acquisition and generalization of new concepts. Both Neches' system and Soloway and Riseman's system make use of prior knowledge in order to tune knowledge. ABLE [Larkin, 1981] uses means-ends analysis to change declarative knowledge into specific patterns or operators.

2.2.2.1 SAGE

Langley [1983, 1987] investigated discrimination as the primary mechanism that underlies skilled problem-solving performance. Discrimination "fixes up" overgeneral knowledge by adding constraints such that the knowledge is restricted to use only in correct situations. SAGE, a system that solves slide-jump puzzles, uses discrimination to learn better operators. SAGE's general operators are tuned to specific patterns resulting in better problem solving performance.

The slide-jump puzzle is played with equal numbers of two types of coins. The coins are initially grouped together with an empty space separating the types of coins, for example:

dime,dime,blank space,quarter,quarter.

The object of the game is to interchange the positions of the two types of coins. In the above example the goal is:

quarter,quarter,blank space,dime,dime.

There are two legal operators, a coin can *slide* into a blank space or a coin can *jump* over a coin of the opposite type into a blank space.

SAGE starts with two general rules, one for sliding and one for jumping. An example of a general rule for the slide operator is:

If a coin is in position1,
    and position2 is to the direction of position1,
    and position2 is blank,
    and that coin can move to the direction,
Then consider sliding that coin from position1 to position2.

SAGE initially solves a slide-jump puzzle using general rules and a trial-and-error method that backs-up whenever a dead-end is encountered. A dead-end is a configuration in which no legal move can be made, for example:

blank-space,dime,dime,quarter,quarter.

SAGE backs-up to the move preceding the dead-end.

As SAGE solves the slide-jump problem, it remembers the successful solution path, disregarding paths leading to dead-ends. On subsequent problem-solving trials, whenever SAGE applies an operator that would take it off the successful solution path, a discrimination mechanism is evoked to learn a better rule. The mechanism compares the situation of the bad application of the operator to the last good application of the operator and constructs a variant of the operator such that it will only apply in the good situation. A variant is created by adding conditions to the rule or changing variables to constants. SAGE creates all possible variants of the rule, and is able to learn multiple conditions and negated conditions. An example of a specific version of the slide operator is:

If a coin is in position1,
    and position2 is to the direction of position1,
    and position2 is blank,
    and that coin can move to the direction,
    and you did not just slide a coin from position2 to position3,
    and you did not just jump the other coin from position1 to position3,
Then consider sliding that coin from position1 to position2.

In Langley's framework, rules have an associated strength which increases with each successful application of the rule. When more than one pattern applies in a particular situation the stronger pattern is preferred. Strength mechanisms are typically used in learning situations where the relevance or irrelevance of particular patterns is not initially specified [Samuel, 1959].

2.2.2.2 LEX

LEX [Mitchell, Utgoff, & Banerji, 1983] is similar to SAGE because it also learns better patterns for specifying the correct application of operators. LEX uses both discrimination and generalization mechanisms to refine these patterns. Mitchell et al. call these patterns heuristics, because they represent when a pattern *should* be used.

A heuristic is represented by what Mitchell et al. call a *version space*. A version space describes a partially learned heuristic by storing a description of its *maximally specific* and *maximally general* elements. The maximally specific boundary describes the most specific version of the heuristic that includes all of the positive training examples seen so far. The maximally general boundary of a heuristic describes the most general version of the heuristic that excludes all of the negative training examples. The plausible descriptions of the heuristic are all the descriptions that fall on or between the two boundaries that delimit the heuristic.

LEX's domain is solving symbolic integration problems. It starts with a problem such as:

$$\int 3x \cos(x)\, dx$$

and uses its current set of operators to get the expression into a form that does not include an integral sign. A set of critics then analyzes a detailed trace of the problem's solution. The critics select a set of positive and negative training instances from the trace. Positive training instances are made up of the steps along the lowest cost solution path. As in SAGE, all steps off the solution path are regarded as negative training instances. LEX uses the information provided by the critics to modify the boundaries of heuristics version spaces. A heuristic's maximally specific boundary is generalized to include all positive training instances. The maximally general boundary undergoes a discrimination process such that all negative training instances are excluded. This discrimination process is similar to that used by SAGE to make its operators more specific such that they only apply in correct situations. The two boundaries that describe a heuristic move towards each other, narrowing the description of the heuristic.

2.2.2.3 HPM

Neches [1987] developed a model of skill acquisition called HPM (Heuristic Procedure Modification). HPM improves performance in a task by gradually transforming an initial strategy into a more efficient form. HPM modifies strategies by employing meta-knowledge in the form of pattern-driven heuristics.

HPM was used to model the development of children's addition strategies. HPM starts with an initial strategy, expressed as a set of production rules, that employ a "SUM" or "counting all" method for addition. The SUM procedure counts out the objects represented by each addend, then these sets are merged into a single set which is counted to produce the sum. The strategy transformation heuristics shown in Figure 2.6 work with a trace record of HPM solving a problem, to gradually transform the addition strategy to a "MIN" or "counting-on" strategy. In a MIN strategy, the larger addend is incremented by the number of times specified by the smaller addend. For example, the *Result still available heuristic* can be applied to the SUM strategy to produce a more efficient strategy. When the SUM strategy is counting up the union set, redundant counting can be eliminated by retrieving the result of counting the first addend. The resulting strategy starts the count up of the union of sets with the value of the first addend and is a step closer to the MIN strategy. This modification produces a more efficient strategy because redundant counting is eliminated.

During the solution of a problem, a goal and production trace record a description of the actions taken to solve the problem. The traces leave a history of the execution and documents the relationships between data and goals. These execution traces provide the detailed information needed by the heuristics to modify both the conditions and actions of operators. Thus, Neches' HPM model of skill acquisition posits the existence of some general optimizing mechanisms, which automatically apply to obtain more efficient solutions.

Result still available:	An item of information has previously been computed, and is needed again.
Modification:	Try to retrieve the item rather than recomputing it.
Untouched results:	A procedure produces an output, but no other procedure receives that result as input.
Modification:	Try to eliminate computation of that item.
Effort Difference:	A difference in expended effort is observed when the same procedure is operating on the same input(s) at different times.
Modification:	Try to use the method involving less effort.

Figure 2.6 Three sample strategy transformation heuristics
(taken from Neches 1987 p.170)

2.2.2.4 Soloway and Riseman's baseball system

Soloway and Riseman's baseball system [1977] learns concepts in the game of baseball by interpretating observed activity in terms of its prior general knowledge about competitive games. The system's prior knowledge is encoded into schemas that drives the interpretation of a baseball game by actively processing new information as an instantiation of its prior knowledge. This type of knowledge-based approach is similar to Rumelhart and Norman's [1978] view of schemas as active knowledge structures that assist in processing and comprehending incoming information.

The baseball system consists of multiple levels that form a taxonomy of patterns and knowledge about competitive games. These levels filter out irrelevant features, form hypotheses about additional features, and construct a hierarchical network that relates the similarities of the acquired concepts. Knowledge is provided at each of these levels in order to aid the processing of information.

The initial levels of the system filter out irrelevant features and partition observations into episodes. The system does this by using general information that identifies relevant features in the observations (e.g., action, actor, location, time) and boundaries of episodes (i.e., a low amount of activity, followed by a high amount of activity, and then another low amount of activity). The filtering process discards any features that do not change from episode to episode. This is essentially a discrimination process that significantly reduces the number of features the system must deal with.

Subsequent levels of processing form hypotheses using prior general knowledge about goals and causal relationships. The description of the baseball game is augmented by hypotheses of goals and causal relationships. For instance, the interaction between a pitcher and batter results in the hypotheses that the goal of the pitcher was to prevent the batter from hitting the ball and the goal of the batter is to hit the ball. These heuristics about goals and causal relationships come from the general competitive game schema.

The final levels of the baseball system partition observations and hypotheses into classes of generalized concepts. This is achieved by using general heuristics which suggest subsets of features that might be useful to the formation of a class. The system knows that the relevant features are: action, actor, location, time, modifiers, goal-of-player, and causal relationship. General heuristics from the competitive game schema dictate which subsets of features to hold constant, while other features are allowed to vary, thus finding similarities and differences within classes. For example, the class "Out" is formed, with "Infield groundout" and "Flyout" forming its subclasses. Both infield groundout and flyout have similar goals, (i.e., to prevent opposition from getting on first-base) but differ in features such as location.

To summarize, Soloway and Riseman's baseball system learns concepts by using a significant amount of highly-organized prior knowledge. This prior knowledge is used to identify relevant features of the observations and infer goals and causal relationships in the game of baseball. The system is able to form generalizations of baseball concepts because the prior knowledge contained in the competitive game schema identifies the few relevant features to generalize across and how these features should be "relaxed" to form more general classes in the taxonomy.

2.2.2.5 ABLE

Larkin's [1981] ABLE system changes its problem-solving strategy from a means-ends analysis method that characterizes a novice, to what Larkin terms the "knowledge development" method of an expert. This is achieved through a learning mechanism that creates specific rules as a result of successful applications of a weak method, means-ends analysis.

ABLE solves kinematics and dynamics physics problems. Its declarative knowledge takes the form of a list of algebraic equations. ABLE starts as a novice using a means-ends analysis strategy to select and solve equations. ABLE's learning mechanism creates a new production whenever means-ends applies an equation to derive a new quantity. The variables of the equation that were known in order to derive the new quantity are put in the condition side of the new production and the newly derived quantity forms the action side. These new productions form highly specialized patterns. To illustrate, if the system solves the following equation for "a":

$$V = Vo + a * t$$

a production is created stating

If V, Vo and t are known then a is known.

This is a highly specific production and will not work, for instance, if "a" is known and "Vo" is not. ABLE will have to create another specialized production in order to deal with this case.

ABLE becomes more expert-like by developing and using new productions that respond to specific situations, instead of using means-ends analysis to search through equations. However, ABLE develops a production for each specific case and does not group these productions according to problem types or any other organization scheme.

2.2.3 Learning larger patterns

The speed with which a particular problem is solved can be improved by learning larger patterns, thereby circumventing the application of many small patterns. The systems discussed in this section all contain mechanisms whose underlying theme is the formation of larger patterns from a series of smaller patterns. These mechanisms include macro-operators [Fikes, Hart and Nilsson, 1972], constructing and debugging procedures [Sussman, 1975], knowledge compilation [Anderson, 1983], and chunking [Rosenbloom and Newell, 1986]. Both Anderson [1983] and Rosenbloon & Newell [1986] propose their mechanisms as psychologically plausible to account for various automaticity effects, such as the speed-up associated with practice.

2.2.3.1 STRIPS

STRIPS [Fikes, Hart & Nilsson, 1972] builds and executes plans that enable a robot to move objects from one room to another. These plans are generalized, stored, and used in future problem solving. As STRIPS gains experience, its repertoire of plans grows and its performance improves because the processing involved in creating plans decreases.

STRIPS constructs plans by finding a sequence of primitive operators that change the world such that its goal becomes true. The sequence of instantiated operators whose actions achieve the goal are gathered together to form a highly specific macro-operator. This macro-operator is then generalized, making it applicable to more situations. Generalization is achieved by changing certain constants into variables. Constants that require specific values in order for the plan to work are not replaced. The generalized macro-operator shown in the triangle table in Figure 2.7 achieves the goal of a robot moving a box into another room. There are two operators in this plan, GOTHRU and PUSHTHRU. The triangle table holds the conditions and actions of the two operators that make up the macro-operator. The predicates in the same row as the operator are the preconditions for the operator. For example, row 2 contains the preconditions of the PUSHTHRU operator: the object to be pushed is in a room, a door connects the room the object is currently in to the room the object will be in, and the robot is in the same room as the object. If these preconditions are true, then the operator PUSHTHRU can be applied, resulting in the robot and the object being in another room. The results of an operators actions appear in the box below the operator.

Macro-operator's can be used in their entirety as an operator in another plan or subsequences of the macro-operator can be used in a plan. For example, if the robot was not in the same room as the object he is required to move, then the whole plan contained in the triangle table of Figure 2.7 would be used. However, if the robot was in the same room as the object then only part of the plan needs to be used, the part that uses the PUSHTHRU operator.

Macro-operators are selected in the same manner as primitive operators. Whenever STRIPS needs an operator, it scans all of its operators for any that have an effect that would allow the achievement of STRIPS goal. Operators, both macro and primitive, do not seem to be organized in any way. The performance improvement occurs primarily because of a savings in the replanning process.

1	INROOM (ROBOT,p2) CONNECTS (p3,p2,p5)	GOTHRU (p3,p2,p5)	
2	INROOM (p6,p5) CONNECTS (p8,p9,p5) CONNECTS (x,y,z) = CONNECTS (x,z,y)	INROOM (ROBOT, p5)	PUSHTHRU (p6,p8,p5,p9)
3			INROOM (ROBOT,p9) INROOM (p6,p9)
	0	1	2

Figure 2.7
A generalized plan in a triangle table

2.2.3.2 HACKER

Sussman's [1975] HACKER system constructed and debugged procedures which solve blocks-world planning problems. HACKER develops from a "novice" with few procedures to solve block-world problems to an "expert" with a well-stocked set of procedures and a collection of errors to avoid.

HACKER learns two things: generalized procedures and generalized bugs. Generalized procedures are sequences of actions to achieve a goal. The following generalized procedure achieves the goal of stacking one block on another after removing everything from the block to be stacked. The lowercase letters preceded by a question mark represent variables.

```
(TO (MAKE (ON ?a ?b ))
        (HPROG (UNTIL ( ?y ) (CANNOT (ASSIGN ( ?y ) (ON ?y ?a )))
                                                (MAKE (NOT (ON ?y ?a )))
                (PUTON ?a ?b ))).
```

English Rendition:

```
TO ACHIEVE ?a ON ?b DO THE FOLLOWING:
        DO UNTIL THERE IS NO ?y ON ?a
                REMOVE ?y FROM ?a
        PUT ?a ON ?b.
```

Generalized procedures are indexed by a pattern constructed from the problem statement. A procedure is found by matching its pattern to the current problem statement. For instance, the above generalized procedure is stored by the pattern "(MAKE (ON ?a ?b))". Generalized procedures summarize and remember the positive results of HACKER's experiences, but do not have any organization.

HACKER generalizes a procedure by changing all possible constants to variables. This contrasts with STRIPS [Fikes, Hart & Nilsson 1972], which is careful not to overgeneralize its macro-operators. HACKER does not have to worry about overgeneralizing because it has the facility to fix "buggy" plans. A generalized bug is a demon that assists in debugging and patching plans. Whenever a generalized procedure cannot execute, a debugging mechanism inspects the procedure to find the problem. The generalized procedure is then "patched". Examples of patches include inserting a new step or rearranging the steps in the procedure in order to avoid the bug. Generalized bugs are constructed by generalizing the information used by the debugging mechanism when it fixes up a procedure. These bugs are used to watch for potentially buggy situations that occur during the construction of new solutions. The following bug was created when the debugging mechanism analyzed and fixed up a procedure that could not execute because of interacting goals (where achieving a goal undoes a previously achieved goal). This bug will watch for plans in which a prerequisite of the second goal will undo a prior goal. In this instance, the goal (ON ?a ?b) is undone by the goal (ON ?b ?c).

```
(WATCH-FOR  (ORDER  (PURPOSE  ?1line  (ACHIEVE (ON  ?a  ?b )))
                    (PURPOSE  ?2line  (ACHIEVE (ON  ?b  ?c ))))
            (PREREQUISITE-CLOBBERS-BROTHER-GOAL
                    ?current-prog  ?1line  ?2line  (CLEARTOP  ?b ))).
```

English rendition:

If a line in a procedure achieves a goal
And another line in a procedure clobbers the first goal
Then call PREREQUISITE-CLOBBERS-BROTHER-GOAL sending it the
procedure, the lines in question, and the prerequisite that is violated.

In the above bug, a patch is applied to the current program by a calling a predefined repair routine called PREREQUISITE-CLOBBERS-BROTHER-GOAL. This routine reorders the offending lines in the procedure. Generalized bugs summarize and remember important negative results of HACKER's experiences, allowing HACKER to learn from its failures.

Whenever HACKER has a blocks planning problem, it searches through its set of procedures for one that matches the problem statement. If a procedure is found, it is used to solve the problem. If a procedure is not found, HACKER constructs one utilizing stored procedures, problem-reduction planning techniques, and information about the blocks world. Constructing a solution is an iterative process, with a set of critics monitoring the developing plan for possible bugs. Critics contain information about all of the generalized bugs. Thus, HACKER's critics, which monitor a developing plan, are more reactive then those found in LEX, which inspect a completed solution. HACKER's generalized bugs represent knowledge about things to avoid in particular situations. HACKER's performance improves because of its ability to utilize past plans and to avoid past mistakes when constructing a plan.

2.2.3.3 Knowledge compilation in ACT Theory

Anderson, Greeno, Kline & Neves (1981) used the ACT theory of skill acquisition to simulate novice-expert differences in geometry proofs skills. This theory of skill acquisition is based on a knowledge compilation mechanism by which performance in a domain becomes more effective and automatic with practice. Knowledge compilation transforms declarative knowledge into procedures and organizes these procedures into packets of related knowledge. Experience enables the system to move from using weak general methods to using powerful domain-specific procedures.

Anderson's ACT theory [1983] clearly separates declarative knowledge from procedural knowledge. The ACT theory assumes that initial knowledge of a domain is obtained from a teacher or a textbook and is encoded in a declarative schema-like form. The side-angle-side postulate (SAS), used in geometry, is shown in Figure 2.8. Its declarative schema-like representation is shown in Figure 2.9. The SAS schema contains slots for background, hypothesis, conclusion, and comments. Information contained in the diagram and statement of Figure 2.8 are included in the background to allow the correct instantiation of the angle and side variables. The hypothesis propositions of the schema must be asserted in previous steps of the proof. They are essentially preconditions that must hold true prior to asserting the conclusion.

This Figure has been removed because of the unavailability of copyright permission.

Figure 2.8 SAS Postulate
(from Anderson et al., 1981)

SAS Schema

Background	s1 is a side of Δ XYZ s2 is a side of Δ XYZ A1 is an angle of Δ XYZ A1 is included by s1 and s2 s3 is a side of Δ UVW s4 is a side of Δ UVW A2 is an angle of Δ UVW A2 is included by s3 and s4
Hypothesis	s1 is congruent to s3 s2 is congruent to s4 A1 is congruent to A2
Conclusion	Δ XYZ is congruent to Δ UVW
Comment	This is the side-angle-side postulate

Figure 2.9
Declarative Representation of the SAS Postulate
(from Anderson et al., 1981)

General procedures interpret and use the declarative knowledge. ACT starts with procedural knowledge that are general interpretive production rules representing weak methods such as means-ends analysis. The declarative information contained in the schema of Figure 2.9 is used by the general procedures of Figure 2.10. The productions represented by P1, P2, P3, and P4 characterize what Anderson et al. regard as the general problem-solving methods that a novice begins with.

```
P1:  IF the goal is to prove a statement
          and there is a schema that has this statement as conclusion
     THEN set as subgoals to match the background of the schema
          and after that to prove the hypothesis of the schema

P2:  IF the goal is to match a set of statements
     THEN match the first statement in the set

P3:  IF the goal is to match a statement in a set
          and the problem contains a match to the statement
     THEN go on to match the next statement in the set

P4:  IF the goal is to match a set of statements
          and the last statement in that set has been matched
     THEN go on to the goal that follows.
```

Figure 2.10
General Productions
(from Anderson et al., 1981)

Knowledge compilation is made up of two components, composition and proceduralization. Composition takes a sequence of production rules that apply one after the other and collapses them into one powerful production rule. This is done by concatenating the conditions and actions of the rules. For instance, if the productions P1, P2, and P3, of Figure 2.10 are applied sequentially, the composition mechanism creates a new production:

```
P5:  IF the goal is to prove a statement
          and there is a schema that has this statement as conclusion
          and the schema has a statement as the first member of its
               background
          and the problem contains a match to the statement
     THEN set as subgoals to match the background
          and within this subgoal to match the next statement of the
               background
          and after that to prove the hypotheses of the schema
```

This new production, P5, will only apply in the same circumstances that would lead productions P1, P2, and P3 to apply and has the same effect as the sequence of productions. All of the conditions in productions P1, P2, and P3 that do not match actions of earlier productions in the sequence are concatenated into P5's conditions. The condition of production P2 is not included as a condition of P5 because it matches the first action of production P1. Similarly, composed productions include all of the actions of the individual productions except for actions that are fulfilled by conditions of a subsequent production in the sequence. The action of P2 is fulfilled by a condition in production P3 and is therefore not included in the actions of production P5. One production, P5, now accomplishes the effects of three productions. Performance is faster because only one production is instantiated and executed, not three. The theoretical assumption is that pattern matching proceeds in parallel and therefore a larger pattern does not take any longer to match than a smaller one.

Composed productions can be combined with other productions that they are executed sequentially with. If production P5 appeared in sequence with two applications of P3, the resulting composed production would be as follows:

```
P6:  IF the goal is to prove a statement
          and there is a schema that has this statement as conclusion
          and the schema has a statement as the first member of its
               background
          and the problem contains a match to the statement
          and the schema has another statement as the next member of
               its background
          and the problem contains a match to the statement
          and the schema has another statement as the next member of
               its background
          and the problem contains a match to the statement
     THEN set as subgoals to match the background
          and within this the next statement of the background
          and after that to prove the hypotheses of the schema
```

The other aspect of knowledge compilation, proceduralization, translates declarative knowledge into procedural form. Proceduralization saves the bindings of variables made during retrieval from long-term memory and encodes these bindings into productions. For example, general interpretive productions, such as P1 through to P6, access and use the knowledge in the declarative schema shown in Figure 2.9. Proceduralization constructs a specific version of a general interpretive production using the information that instantiated the general rule thus eliminating the need to retrieve and match declarative knowledge. The result of production P6 matching the information contained in the SAS schema of Figure 2.9 is:

```
P7:  IF the goal is to prove Δ XYZ is congruent to Δ UVW
          and S1 is a side of Δ XYZ
          and S2 is a side of Δ XYZ
          and A1 is an angle of Δ XYZ
     THEN set as subgoals to match the background of the SAS schema
          and within this to match the next statement in the schema
          and after that to prove the hypotheses of the schema
```

The next time the situation occurs, the more specialized pattern is preferred, thus eliminating retrieval of variable bindings from long-term memory. Performance is faster because the declarative information forms a specific pattern in the rule.

Composition and proceduralization interact to create large, specific rules such as:

```
P8:  IF the goal is to prove Δ XYZ is congruent to Δ UVW
          and S1 is a side of Δ XYZ
          and S2 is a side of Δ XYZ
          and A1 is an angle of Δ XYZ
          and A1 is included by S1 and S2
          and S3 is a side of Δ UVW
          and S4 is a side of Δ UVW
          and A2 is an angle of Δ UVW
          and A2 is included by S3 and S4
     THEN set as subgoals to prove
          S1 is congruent to S3
          S2 is congruent to S4
          A1 is congruent to A2
```

The knowledge contained in the declarative schema of Figure 2.9 is now completely encoded in production P8. In the ACT framework, the knowledge compilation mechanisms, proceduralization and composition, are posited to account for the automaticity that characterizes skill acquisition.

2.2.3.4 Chunking

Chunking was proposed as a general learning mechanism by Rosenbloom and Newell [1986], primarily to account for the power law speed-up that results from practice.

This mechanism acquires rules or chunks as a result of its goal-based experiences. Chunks are patterns that are created by processing the goals of a task. The condition side of the chunk is formed from the parameters of a goal, and the action side of the chunk is formed from the result of a goal. Chunking enhances performance by replacing the processing of a goal and its subgoals by a condition-action pattern. In this respect, it is quite similar to compilation.

The three components that make up a chunk are the encoded parameters, the encoded results, and the connection between these two components. Parameters of a goal are encoded into a new object that represents their combination. The results of a goal are also represented by an encoded object. The connection component ties the encoded object that represents the goal's parameters to the encoded object that represents the goal's results. Two conditions must be met to create a chunk for a goal. First, the goal must have completed successfully and second, all of its subgoals must be chunks. A chunk is used when the connection component matches the encoded parameter object and its related active goal, thereby decoding the results of the goal. This can be thought of as a production rule that responds to an active goal and an object that represents the goals parameters. The action side of the rule decodes the object that represents the results of the goal.

The following example, taken from Rosenbloom and Newell [1986], illustrates the process of chunking. Figure 2.11 shows a simple goal hierarchy for the averaging of two numbers. The top goal is Compute-Average-Of-Two-Numbers, which takes two numbers as parameters and returns their mean. Compute-Average-Of-Two-Numbers is decomposed into two subgoals, Compute-Sum-Of-Two-Numbers and Divide-Sum-By-2. Compute-Sum-Of-Two-Numbers takes two numbers and returns their sum. Divide-Sum-By-2 takes the previously calculated sum, divides it in two, and returns it as a result.

This Figure has been removed because of the unavailability of copyright permission.

Figure 2.11
A simple three-goal hierarchy for the averaging of two numbers. (From Rosenbloom and Newell, 1986)

Suppose that the system was given the task "Find the average of 3 and 7". This goal's parameters are the numbers 3 and 7. While the system solves this task, using the numbers 3 and 7, two chunks are created. Control passes from goal 1 to goal 2 and when goal 2 has completed successfully, a chunk is created for it. This chunk is shown in the bottom left-hand side of Figure 2.12. The parameters 3 and 7 are encoded into the object E1. A decoding object is created, D1, to decode the result of the goal into the value 10. The connection made between E1 and D1 specifies the goal that was chunked. D1 is decoded whenever this connection component detects the presence of goal 2 and the encoded parameter E1. This can be interpreted as rules such as:

If 3 and 7 are present then make the object E1 active.

If E1 is active and goal 2 is active then activate D1.

If D1 is active then add 10 to working memory.

This Figure has been removed because of the unavailability of copyright permission.

Figure 2.12
Sample chunks created for the hierarchy of Figure 2.11
(From Rosenbloom and Newell, 1986)

After goal 2 has completed successfully, control returns to goal 1 and then control moves to the other subgoal, goal 3. A chunk is created for goal 3 when it completes successfully. This chunk is shown in the bottom right-hand corner of Figure 2.12. The chunk encodes the number 10 into the object E2 and decodes the object D2 into the number 5. These objects are connected by a component that responds to the presence of goal 3 and the encoded parameter E2. This chunk is more flexible than the chunk made for goal 2 because it responds to any situation in which goal 2 returns a result of 10. The chunk formed for goal 3 is not concerned with the parameters that generated the number 10, they could have been 3 and 7 or 2 and 8. The task is complete after goal 2 terminates. Control is passed back to goal 1 but a chunk is not created for it because its subgoals were not processed by chunks.

The second time this same task is performed, goal 1 is activated and the values 3 and 7 are put in the system. The chunk for goal 2 encodes these values and replaces them with the object E1. Control passes from goal 1 to goal 2 and because both goal 2 and E1 are present D1 becomes active and is decoded to the value 10. As soon as the value 10 is detected the chunk for goal 3 encodes this value into the object E2. Goal 2 has finished therefore control is passed back to goal 1 and then to subgoal 3. Because goal 3 is active and E2 is present, D2 becomes active and is decoded to return the value 5. Both of goal 1's subgoals were processed by chunks, therefore a chunk is created for goal 1. The encoding component of this chunk encodes the parameter E1 into the new object E3. This is an example of what Newell and Rosenbloom call "hierarchical encoding": E3 encodes E1 which encodes the actual parameters of 3 and 7. Objects that encode parameters can become parameters of other goals resulting in an encoding hierarchy. Similarly, the decoding component of the chunk decodes D3 to D2 which then decodes to the number 5. A decoding hierarchy is formed when complex results are decoded to simpler results which are themselves decoded to even simpler results.

The third time this task is performed, E1 will be encoded as a result of the parameters 3 and 7. E3 will then immediately be encoded as a result of E1. Because goal 1 is active and E3 is present the connection is made to D3 and goal 1 is completed without ever having to execute its subgoals. If the result of D3 is required, it will be decoded to D2 which is then decoded to the number 5. Performance improves because the high-level goal is directly processed using a pattern recognition rule reducing the need to decompose the high-level goal into subgoals. This corresponds to the savings associated with knowledge compilation.

2.2.4 Learning New Organizations of Patterns

A number of researchers have developed mechanisms for reorganizing existing knowledge to assist and improve problem solving and understanding. All of these systems can be viewed as learning by restructuring, but there are cases in which the reorganization mechanisms themselves functionally serve the same purposes as generalization and discrimination. Lebowitz [1983] developed a system called IPP, that uses its organization of past experiences to understand terrorist attack stories, e.g. make valid inferences about type of activites. Lebowitz [1986] also uses this framework to learn concepts. Riesbeck

[1984] uses reorganization-of-knowledge to form a model of expert and novice reasoning differences in an economic domain. Kolodner and her associates have constructed a number of systems that become better problem solvers by re-organizing experiences. SHRINK [Kolodner, 1983] learns to analyze new psychiatric cases, MEDIATOR [Kolodner, Simpson, & Sycara-Cyranski, 1985] interprets and resolves disputes, and JULIA [Kolodner, 1987] is a catering system that plans meals for clients. In these systems, the problem solver relies on its memory of past experiences to guide solution of the current problem and to avoid failures that occurred in previous problems.

All of the above systems are based on theories of memory developed by Schank [1980] called Memory Organization Packets (MOPs). I will describe only Kolodner's systems in more detail, but most of the main ideas characterize Lebowitz's and Riesbeck's systems as well.

In Kolodner's systems, experiences are stored in an E-MOP (Episodic Memory Organization Packet) structure, which is similar to a discrimination network [Feigenbaum & Simon, 1984]. E-MOPs are nodes in the network. These nodes are complex data structures which hold generalized knowledge common to the individual experiences the node organizes. This generalized knowledge includes solutions and failures encountered in past experiences. New experiences are stored under the nodes according to features that differentiate them from other experiences stored under the node. Connections between the nodes of the network represent the differences encountered in experiences.

The problem solver retrieves information from the E-MOP network to assist it in the solution of the current problem. Features of the current problem direct the problem solver to areas of the network that store similar problems. The solutions of similar past experiences can be used to construct a solution to the current problem. A feature of the current problem can also lead to a similar past experience that involved a failure during its solution. Information stored about the type of failure, the reason for the failure, and the solution to the failure, can be used to assist the current problem avoid a similar failure.

The learning mechanisms in Kolodner's system are the mechanisms that maintain the organization of experiences in memory. These mechanisms incorporate a new experience in the network as a function of its similarities and differences to the existing experiences in the network. When a new experience has a common feature with a past experience, similarities between the two experiences are stored on a newly-created E-MOP. This is essentially a generalization process in which stereotypical features are abstracted. Experiences are organized under a given E-MOP according to their differences. This is essentially a discrimination process, since these differences (which connect two MOPs) are constraints that must be met in order to travel to the next E-MOP. Experiences are also used confirm and refine the generalized information stored on E-MOPs. Unlike other learning systems, all individual experiences are stored. Because all experiences are retained, commonalities can be abstracted between experiences that do not occur sequentially in time.

As experience is gained, more cases are stored in the network giving the problem solver more solutions to draw upon. Features of the current problem direct it to E-MOPs about similar problems. These E-MOPs may contain information useful for the solution. These features can also direct the problem to E-MOPs that record information about failures. Information about the cause of the failure and the remedy to the failure can be used to avoid similar failures. This kind of information, stored on the E-MOPs, allows shortcuts in problem solving and avoids detailed reasoning and failure processes.

2.2.5 Summary

In Rumelhart's and Norman's [1978] framework for learning, tuning changes the existing content of a schema to better cope with the functional demands made on the schema. Discrimination, generalization, strategy transformation heuristics, and proceduralization are all methods that tune existing knowledge. The parameter encoding and decoding features of Newell and Rosenbloom's [1986] chunking mechanism and the mechanism used in ABLE [Larkin, 1983] both correspond to the proceduralization aspect of Anderson et al's. [1981] knowledge compilation process. All three of these mechanisms create more specific patterns by putting the retrieved (declarative) knowledge directly into some procedural form.

Restructuring, in Rumelhart and Norman's framework, modifies the organization of knowledge structures and occurs when new information does not fit existing schemas or when a group of schemas needs reorganization. The general theme of "forming large patterns of conditions and actions from a series of smaller patterns of conditions and actions" characterizes the process of macro-operators, generalized procedures, chunking, and composition. These mechanisms can be viewed as a weak form of restructuring. Each of these mechanisms notices regularities in the application of knowledge and creates new schemas that group together sequences of operators. These mechanisms exhibit the behaviour attributed to schema induction [Rumelhart and Norman, 1978].

There are, however, some important differences between macro-operators, chunking, and composition. In Andersons's [1983] framework, composite productions are more general than chunks because they retain variables whereas Newell and Rosenbloom's [1986] chunks are made up of constants. In other words, Anderson's composition and proceduralization steps are quite distinct, whereas in Newell and Rosenbloom's framework, one might claim that composition and proceduralization are occurring more in tandem. Composite productions have an implicit set of matching constraints imposed on how variable bindings will be done. For example, a production with a variable =x appearing in several of its conditions must all match the same constant. Both chunks and composite productions differ from macro-operators because the individual components making up the chunk or production cannot be accessed whereas the subcomponents of a macro-operators can be.

Discrimination, generalization, strategy transformation heuristics, macro-operators, proceduralization, composition, and chunking concentrate on the organization and structure of operators or groups of operators. These mechanism are not concerned with the organization of operators and knowledge *relative* to one another. In contrast, Lebowitz [1983], Riesbeck [1984], and Kolodner and her associates [1983, 1985, 1987] impose an organization on knowledge and address issues of knowledge reorganization. These systems contain mechanisms that tune and restructure knowledge in the network as new experiences are indexed. New schema are created through similarities to existing schema (patterned generation) and by noticing regularities in knowledge (schema induction). These systems learn new organizations of knowledge, and in some instances, new concepts with experience.

Although Soloway & Riseman's baseball system [1977] and Kolodner's E-MOP framework [1984] seem similar because they both form a hierarchial network of concepts, there are some important differences between these systems. Kolodner's E-MOP framework is more syntactic than Soloway and Riseman's baseball system because it does not contain as much prior knowledge. Kolodner's E-MOP framework learns the relevant discriminating features by organizing knowledge according to its similar and different features. In contrast, Soloway and Riseman's system knows what the relevant discriminating features are about baseball from its general competitive game schema. Also, the internal structure of an E-MOP and the organization of E-MOP's relative to one another changes in Kolodner's system. Levels of generality in Soloway and Riseman's system are fixed by the general competitive game schema, which further dictates how new baseball concepts will be constructed.

Chapter 3

A Framework for Knowledge Reorganization

3.1 Introduction

There are two key empirical results that I am interested in simulating. The first concerns differences in knowledge organization and the second concerns the different problem-solving strategies used by novices and experts. As noted in the previous chapter, experts seem to organize their knowledge around what might be called "deep features" or underlying principles [Chi, Feltovich & Glaser, 1981]. In contrast, novices seem to organize their knowledge around "surface features" typically found in problems. With respect to general problem-solving strategies, novices tend to reason backwards from the desired quantity, while experts reason forward from what is known [Simon & Simon, 1978; Larkin et al., 1980]. I believe that the changing organization of knowledge contributes to the change in problem-solving strategies. Solution methodologies that build up with experience may enable more opportunities for reasoning forward. While Chi et al. [1981] and Larkin et al. [1979, 1980] have characterized the endpoints of novice and expert behavior, what is missing is a psychologically plausible process model that takes us from the novice endpoint to the expert endpoint.

Although some of the learning mechanisms reviewed in the previous chapter can support a shift in problem-solving strategies of novices and experts, most do not provide a way of qualitatively restructuring knowledge around different problem types of a domain. Forming macro-operators and generalized procedures, knowledge compilation, and chunking all group sequences of operators to avoid recomputation in future problem solving. These mechanisms organize operators according to their occurrence in time (i.e., sequentially) and not according to their features or underlying structure. Proceduralization, generalization, and discrimination refine pieces of knowledge as a function of experience, but cannot support a broader restructuring and reorganization of these individual pieces of knowledge. The exception is the framework used by several researchers, Lebowitz [1983], Riesbeck [1984], and Kolodner and her associates [1983, 1985, 1987], in which the organization of knowledge dynamically changes as a function of experience. The general process model I propose for the shift in problem-solving strategies as well as knowledge organization is based on Kolodner's [1984] theory of reconstructive memory.

Kolodner's [1984] model of reconstructive memory organizes events into categories in memory. These conceptual categories are referred to as Episodic Memory Organization Packets (E-MOPs). An E-MOP structure is very similar to a large EPAM-like discrimination net [Fiegenbaum & Simon 1984] in which the nodes of the network are complex knowledge structures. The knowledge in the net is organized according to its similarities and differences. A node is either an event or a representation of similarities among more than one event. Differences among events are indices to other nodes. The E-MOP framework provides a method of organizing, storing, and retrieving large amounts of complex, interrelated information.

Lebowitz [1983], Riesbeck [1984], and Kolodner and her associates [1983, 1985, 1987], framework for maintaining a "useful" organization of events in long term memory can be viewed as a type of learning system. In particular, it has considerable theoretical overlap with Rumelhart and Normans [1978] notions of learning as accretion, tuning, and restructuring. What Kolodner [1984] has focused on, from a memory organization viewpoint, are mechanisms that essentially restructure existing knowledge. Furthermore, the "recognition" of similarities and differences among nodes in the network corresponds to basic ideas of generalization and discrimination. Most importantly, what is regarded as a useful organization of knowledge - the correct set of similarities and differences - changes with experience. E-MOP mechanisms dynamically reorganize the network as new knowledge is added, changing the content of the nodes (similarities) and the nature of the indices (differences).

3.1 Overview of the CYRUS system

Kolodner's [1984] system, CYRUS, organized events about Cyrus Vance's diplomatic meetings. The following examples are taken from her description of the system. Three events are shown in Figure 3.1. The E-MOP organization of this information is shown in Figure 3.2.

EV1 Meeting is political and occupational for Vance.
Participants = Vance and Begin.
Participants = foreign diplomats.
Topic is an International contract.
Topic involves United States.
Begin's nationality = Israeli.
Begin's occupation = head of state.
Topic's sides = Israel and Arabs.
Topic = Arab-Israel peace.
Underlying topic = peace.

EV2 Meeting is political and occupational for Vance.
Participants = Vance and Gromyko.
Participants = foreign diplomats.
Topic is an International contract.
Topic involves United States.
Gromyko's nationality = USSR.
Gromyko's occupation = foreign minister.
Topic's sides = USSR.
Topic = SALT.
Underlying topic = arms limits.

EV3 Meeting is political and occupational for Vance.
Participants = Vance and Sadat.
Participants = foreign diplomats.
Topic is an International contract.
Topic involves United States.
Sadat's nationality = Egypt.
Sadat's occupation = head of state.
Topic's sides = Israel and Arabs.
Topic = Arab-Israel peace.
Underlying topic = peace.

Figure 3.1 Sample events.

Inspection of the three meetings of Figure 3.1 reveals that all three have some commonalities and that EV1 and EV3 are similar in all but one feature. In Figure 3.2, MOP1 is a packet of information that records commonalities and differences between all three events indexed in the network. All three meetings were a political and occupational activity for Cyrus Vance, had foreign diplomats as participants, and involved a topic of an international contract that included the United States. Because these features occurred in all events stored under MOP1, they are the *norms* of MOP1. Norms are features which are common to the events that a particular E-MOP organizes (i.e., events that are stored "under" the E-MOP in the network). The differences among the three events involved the participants and the topic of the meetings. These differences convey more specfic information about the events. For example, the norms of MOP1 say that participants are foreign diplomats but the differences specify that the participants are from Israel, Egypt, or the USSR and that their occupations are either foreign minister or head of state. The three events indexed under MOP1 differ in their topic, topic's sides, underlying topic, participant's occupation, and participant's nationality.

EV1 and EV3 have many features in common and as a result four of their differences from the norms of MOP1 are the same. In both meetings, the participants were heads of state, the topic's side included Israel and the Arabs, and the topic was Arab-Israel peace, with an underlying topic of peace. The only difference between EV1 and EV3 is the participant's nationality. MOP2, MOP3, MOP4, and MOP5 record the commonalities and differences between EV1 and EV3.

Arcs between E-MOP's and their sub-MOPs and events are called *indices*. The indices of an E-MOP correspond to differences in the events that the E-MOP organizes. Indices are features with a type and a value. In Figure 3.2, the feature "participants occupation" has a value "foreign minister" to form the complete index "participants occupation = foreign minister" which leads to EV2. Indices lead either to individual events or to other, more specialized E-MOPS. These more specialized E-MOPs are simply a further organization of a subset of events with a common indexing feature. An important feature of this framework is that multiple indices can lead to the same knowledge. These structures provide a rich cross-indexing of knowledge through multiple retrieval paths.

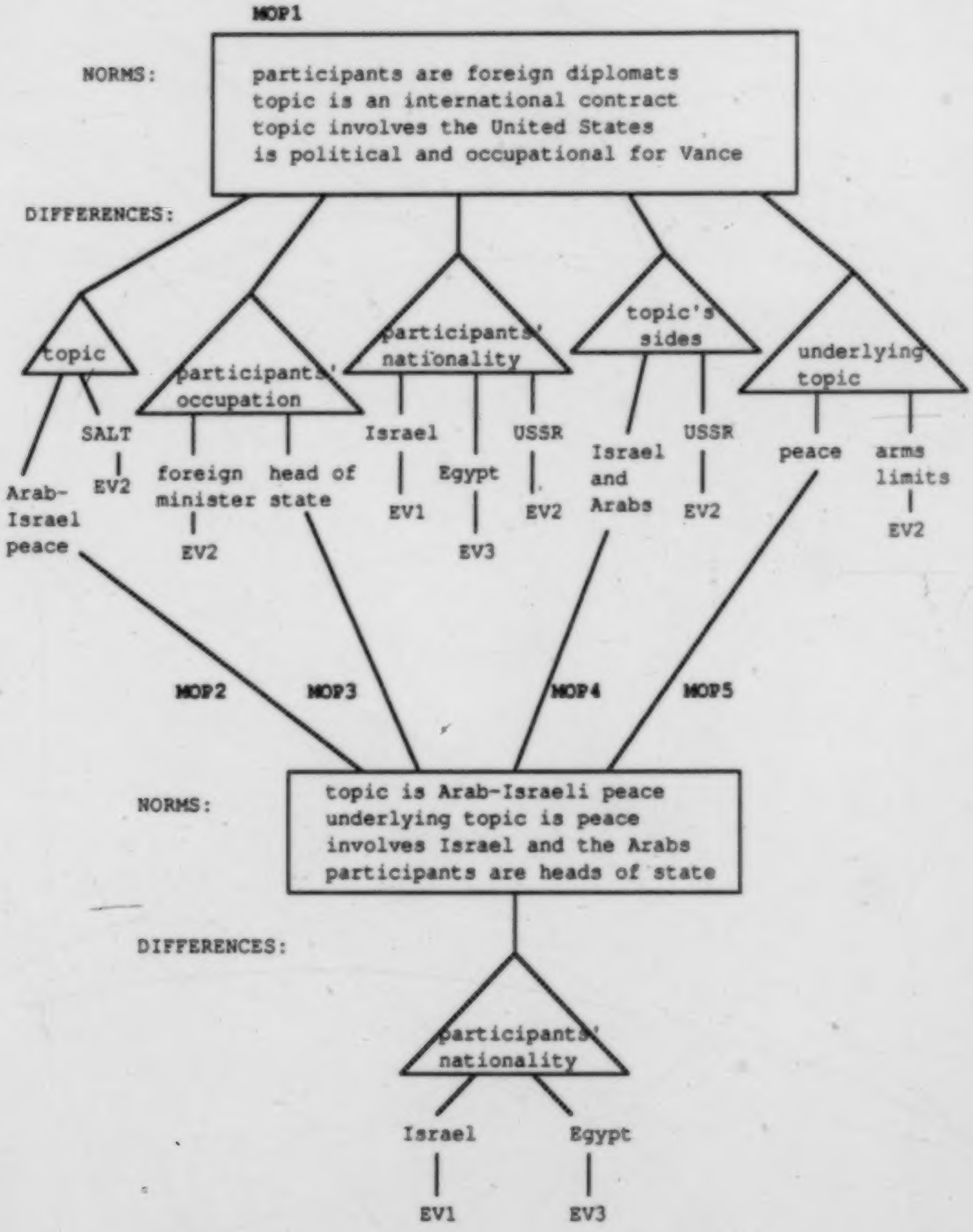

MOP1
NORMS:
participants are foreign diplomats
topic is an international contract
topic involves the United States
is political and occupational for Vance
DIFFERENCES:
topic
Arab-Israel peace
SALT
EV2
participants' occupation
foreign minister
EV2
head of state
participants' nationality
Israel
EV1
Egypt
EV3
USSR
EV2
topic's sides
Israel and Arabs
USSR
EV2
underlying topic
peace
arms limits
EV2
MOP2
MOP3
MOP4
MOP5
NORMS:
topic is Arab-Israeli peace
underlying topic is peace
involves Israel and the Arabs
participants are heads of state
DIFFERENCES:
participants' nationality
Israel
EV1
Egypt
EV3

Figure 3.2 An E-MOP structure

An E-MOP network represents an organization of knowledge according to a particular scheme of similarities and differences. The adequacy of this organization is confirmed or disconfirmed by subsequent events. In Figure 3.2, the index "underlying topic = peace" leads to MOP5. MOP5 states that when the underlying topic is peace, then the topic is Arab-Israeli peace, which involves heads of state from Israel and the Arabs. This information may not be true of later events. For instance, if subsequent events indexed under MOP5 are like EV4 (with an underlying topic of peace but involving other countries and other specific topics), then the generalized information of MOP5 must reflect this.

```
EV4    Meeting is political and occupational for Vance.
       Participants = Vance and Gromyko.
       Participants = foreign diplomats.
       Topic is an International contract.
       Topic involves United States.
       Gromyko's nationality = USSR.
       Gromyko's occupation = foreign minister.
       Topic's sides = USSR.
       Topic = Afghanistan invasion.
       Underlying topic = peace.
```

In other words, what was thought to be common to all events with "underlying topic = peace" turns out to be wrong. Any incorrect commonalities, called *false norms*, must be removed. Similarly, there may be a problem in the indices of E-MOPs. If all subsequent events indexed in the E-MOP structure of Figure 3.2 had the feature "participants occupation = head of state", then this feature should be removed as an index (a difference) and put on MOP1 as a norm (a similarity) because it is generally true of all events indexed under this E-MOP. The index "participants occupation = Foreign minister" would remain because it is a difference from what is normally true of the events indexed under MOP1.

3.3 How E-MOPs are dynamically reorganized

A new event is integrated into the E-MOP network by following indices that match the new event's features. Both the type and value of an index must be a feature of the new event for the index to be followed. In this way, an index is a kind of "test" that must be passed in order to arrive at a particular node in the network. Event 4, as shown above, would be indexed in the E-MOP network of Figure 3.2 in the following way. The E-MOP network includes the indices "topic's sides = USSR", "participants nationality = USSR", and "participants occupation = foreign minister". These are features of this new event, and all of them lead to event EV2 in the E-MOP network. The index "underlying topic = peace" leads to MOP5. No further traversal from MOP5 is possible because the indices out of MOP5 specify that the participants' nationality must be either Israeli or Egyptian, but in EV4, the participants' nationality is USSR. Having "arrived" at one or more places in the network, the new event is then integrated into the network. As new events are stored in the network, information is changed on existing E-MOPS and new E-MOPS and indices are created. The following sections describe the basic mechanisms used by Kolodner to maintain memory organization.

3.3.1 Index Selection

Indices are used to discriminate individual events and divide a category into reasonably-sized subcategories. It is important to find indices that will fulfill these requirements to constrain the potential combinatorial explosion of indices in the network. As an event traverses the E-MOP network, indices are chosen based on the information contained on the E-MOP that the event has reached. Initially, all of the features of the incoming event are regarded as potential indices. The norms of an E-MOP are eliminated from this list of potential indices because they are true of all events and therefore are not good discriminating features among events. For example, if EV4 were indexed into the E-MOP structure of Figure 3.2, then the features in EV4 that occur as norms in MOP1 would be eliminated as potential indices. This leaves the last five features of the EV4 description as potential indices. Three of these features, "participants nationality", "participants occupation", and "topic's sides" lead to EV2. One feature, "underlying topic", leads to

MOP5. The remaining feature "topic = Afghanistan invasion" does not exist as an index in the network. Therefore, this index is created and EV4 is stored with this index under MOP1 and MOP5.

As events are stored in the E-MOP network, the system records information about the features that make good indices and the features that do not. A feature which is idiosyncratic and does not co-occur with any other features will not be a good index because it will not constrain indexing. This is called a *non-predictive* feature. In contrast, a feature that co-occurs with other features is considered to be a good index. This is because whenever this feature appears it can be inferred that its co-occurring features are probably present. This is called a *predictive* feature. As more events are indexed, features may be tagged as non-predictive and will not be eligible as indices in the future.

3.3.2 Creating an E-MOP

New E-MOPs are created where ever multiple events are indexed under the same feature. When an event traversing an E-MOP network arrives at a node where an event is stored, a specialized E-MOP is created. The norms of the new E-MOP are based on the similarities between the events it indexes. The feature indexing the E-MOP is also a norm. Events are indexed under the new E-MOP according to their differentiating features. For example, in Figure 3.2, MOP2 was created because both EV1 and EV3 shared the indexing feature "topic = Arab-Israel peace". The four common features of the two events are generalized and held as norms on MOP2. The only difference between the two events is the value of participants' nationality and this is the index that leads to these two specific events. Currently, the four sub-MOPs of Figure 3.2 are identical because they were all created from the same two events. However, as subsequent events are indexed in the network, it is possible that these four E-MOPs will become different from each other. This is because an event (i.e., EV4) may traverse a subset of the four indices leading to these sub-MOPs. Only the sub-MOPs that are traversed will be changed by the event.

3.3.3 Monitoring the E-MOP Network

The E-MOP network is monitored and changed as new events are indexed into the network. As events are indexed in an E-MOP, a commonality may show up that is a feature of most events organized by the E-MOP but is not one of the E-MOP's norms. In other words, a feature that should be a norm (a commonality) on this E-MOP is instead an index (a difference). Because the feature is true of most events, the difference leads to a sub-MOP that is indexing a majority of its parents events. When this occurs, the sub-MOP is *collapsed* into the parent E-MOP. This means all norms of the sub-MOP are moved up to the parent and the sub-MOP is destroyed. Thus, a difference that served as an index to a sub-MOP now becomes a norm (commonality) of the parent E-MOP. This process of monitoring and collapsing E-MOPs causes norms that are common to many events to move "higher" in the network, because they rightly characterize a large subset of the knowledge in the network.

Conversely, an E-MOP might contain norms that, over time, are not shared by a majority of events indexed "beneath" it. Whenever an event arrives at an E-MOP, the event is used to check the E-MOP's information. Norms of the E-MOP that are not true of the current event are removed and a sub-MOP is created for each of them. These over-generalized features characterize a smaller subset of events than was originally believed and are moved "lower" in the E-MOP network.

3.3.4 Issues in controlling the mechanisms

Kolodner uses a number of thresholds to control the E-MOP restructuring mechanisms of the model. An E-MOP becomes *stable* once it indexes a predefined number of events. A sub-MOP that indexes a majority of its parents events can only collapse into a stable parent E-MOP. Each norm of an E-MOP has a *certainty* associated with it. A norm's certainty is increased with every event that has this norm as a feature, and decreased with every event which does not contain the feature. When the norm reaches a predefined upper threshold, the norm is regarded as *actual* and can no longer be removed from the E-MOP. A norm that reaches the predefined lower threshold is removed from the E-MOP. However, an E-MOP must be stable before the information contained on the E-MOP can be changed. This means that information on an E-MOP cannot move up or down in the network until a semblance of stability is attained.

3.4 Summary

To summarize, the E-MOP framework can be viewed as a complex discrimination network of nodes, where nodes represent collections of features common to the nodes "beneath" it in the net. Paths between the nodes are "differences" that further distinguish between the information contained in the network. The paths can be thought of as "tests" to be passed before access can be gained to other areas of the network. Multiple retrieval paths (indices) leading to the same E-MOP provides a rich cross-indexing of knowledge. An important aspect of E-MOPs is that their organization responds to experience. Mechanisms tune and restructure the E-MOP network to accommodate incoming knowledge.

It is important to distinguish an E-MOP from a schema or frame, where the latter structures are typically complete units of knowledge about some concept. I view a set of E-MOPs as a generalized, but incomplete, pattern. Its associated indices can be viewed as constraints on further parts of a pattern. In this way, knowledge is *distributed* over a number of levels of E-MOPs, which together might be functionally equivalent to a schema.

Chapter 4

The EUREKA System Design

The goal of the EUREKA system is to model two important aspects of novice-expert behavior and to explore a set of mechanisms that underlie the transformation of the novice to an expert behavior. This chapter describes the design of EUREKA, focusing on mechanisms that restructure knowledge in order to simulate the novice-to-expert transition in both knowledge organization and problem-solving strategies.

4.1 Overview of the model

EUREKA contains three major subcomponents: "unorganized physics knowledge", knowledge organized in a P-MOP (Problem Memory Organization Packets) network, and a problem solver. The model assumes that the novice has a good deal of unorganized facts about physics. The unorganized knowledge reflects my assumptions about the basic facts a novice might know about physics from studying the textbook. For example, I assume that novices could give an equation for Newton's second law even if they could not immediately recognize its applicability to a particular problem. This physics knowledge is stored in a semantic network and takes the form of concepts, inferences, and equations about Newton's force laws and conservation of energy.

The organized knowledge stores EUREKA's problem-solving experiences in a P-MOP network. P-MOPs organize knowledge around commonalities and differences and dynamically change their structure as new information comes in. In EUREKA's novice stages, P-MOP knowledge is grouped around the surface features encountered in problems. This early stage contains few solution methodologies and inferences. As the organized knowledge base evolves with experience, it reflects a more expert-like organization. The knowledge base becomes organized around physics principles and contains solution methodologies and inferences. The solution methodologies are represented by goal-subgoal hierarchies and are viewed as compiled methods for achieving a particular goal. This means that solution methods can only be accessed by their top goal and are therefore used in an all-or-none manner. Although subportions of a solution method are not available for use, portions of the goal-subgoal hierarchy that are common across methodologies are abstracted and moved up in the P-MOP network.

Mean-ends analysis is at the heart of the problem solver. It initially works much the same as Larkin's [1981] model. The means-ends problem solver solves the problem using the unorganized physics knowledge. The problem solving experience is then stored in the P-MOP network. An essential difference is that EUREKA consults the knowledge organized in the P-MOP network prior to using means-ends analysis to solve the problem. Relevant inferences or methodologies that are found in the P-MOP network are used to assist in the problems solution. Features of the current problem are used to direct the retrieval of information from the relevant parts of the P-MOP network. Moreover, the information contained in the P-MOP network that has been most frequently matched will be retrieved first. As EUREKA solves more problems it uses its previous experiences and relies less on means-ends analysis.

Before exploring the three components of the model in detail, I will present a sample problem representation prior to it being solved as well as after its solution. I then describe a P-MOP that indexes this problem.

4.2 How a problem is represented

The representation for a physics problem given to EUREKA initially contains a group of predicate-value pairs that describe the problem statement. Figure 4.1 shows the initial problem representation for the following problem, problem1:

> A body of mass 2 slugs is on an inclined plane that has an angle of 30 degrees from the horizontal. The coefficient of friction for the plane is 0.3. Find the acceleration of the body.

A problem representation is organized around objects, features of objects, and relationships between objects. It also includes the known and desired quantities of the problem and the forces due to gravity that act on the body. In this problem, the reference frame is specified such that the x-axis is along the inclined plane in the direction of motion and the y-axis is perpendicular to the x-axis. This problem is given to EUREKA as the following problem description.[1]

```
Objects     body-1, surface-1
Relations   on(body-1 surface-1)
Features    moving(body-1,time2), inclined(surface-1),
            friction(surface-1), mass(body-1)
Knowns      (body-1 mass), (surface-1 incline-angle), gravity,
            coefficient-friction, (body-1 acceleration-along-y-axis)
Desired     (body-1 acceleration-along-x-axis)
Forces      (body-1 gravity-along-x-axis),
            (body-1 gravity-along-y-axis),
```

Figure 4.1
Initial problem representation of Problem1

EUREKA does not derive its own representation of the problem from the original English description. Novak's [1977] system ISAAC addressed this very difficult job of transforming an English problem description into a representation of the problem's objects, features of objects, and relationships among objects. ISAAC's main contribution was its ability to recognize how objects, their features, and relations to each other all instantiated what Novak called "canonical physics objects". Examples of canonical physics objects are "lever", "pivot", or "point mass". The view of an object as a particular canonical physics object depends on the object's context in the problem statement. For example, a person carrying a board can be viewed as a pivot whereas a person sitting on a board can be viewed as a point mass. ISAAC's final representation of the English problem statement described the problem as a function of these canonical objects, their features, and their relations to each other. EUREKA's input is similar to ISAAC's final representation of the English problem statement, except that EUREKA does not have concepts like canonical objects (the ability to abstract a set of defining characteristics of canonical objects is a potential side-effect of EUREKA's learning that is discussed later).

---

[1] The initial problem representation and the frame of reference are areas in which expert-novice differences can occur as well, but I ignore them for this study.

During the course of problem solving, EUREKA's problem representation becomes "enhanced" with inferences, facts, and solution methodologies. An inference is a rule of the form antecedent implies consequent. It is this enhanced problem representation that is subsequently indexed into the P-MOP network. Figure 4.2 shows the representation that emerges after problem1 has been solved. During the solution of this problem inferences are retrieved from the semantic network (the mechanisms by which this is done are discussed later). If the conditions of an inference are present as features in a problem, then the representation is augmented with the inference, and the conclusion of the inference. For example, the inferences in Figure 4.2 state that normal and frictional forces are acting on the body. The conditions of the inferences appear as features in the problem representation. I will refer to the results of these two inferences as "abstract" physics features.

The second type of new information in the representation is "derived quantities." These result from using equations with known quantities. In this example, seven quantities are derived. The enhanced representation also contains a solution method. A solution method is a group of subgoal lists that record the order in which EUREKA focused on various concepts during solution and the equations that involved these concepts. The solution method in Figure 4.2 is redrawn in Figure 4.3 as a tree structure. The order in which EUREKA focused on concepts in this problem is as follows: acceleration along x-axis, sum of forces along the x-axis, gravity along the x-axis, the relationship "on", friction along the x-axis, normal force along the y-axis, sum of forces along the y-axis, and gravity along the y-axis. For each focus-of-attention, EUREKA retrieves equations in which the focus appears as a variable. The equations EUREKA uses to solve for a concept are listed under the concept in Figure 4.3 (concept "on" has no associated equations). All information contained in the final, enhanced problem representation is used when storing the problem-solving experience in the P-MOP network.

Enhancing a problem representation can be thought of as moving through "stages of representation" as described by Larkin [1985]. Although EUREKA does not go through these stages explicitly, this enhancement process is an important kind of transformation. The problem that is ultimately indexed contains many new concepts and quantities that are inferred during solution. While in its novice stage, EUREKA solves a problem by immediately generating equations using mean-ends analysis. During this period, inferences gathered from EUREKA's past problem-solving experiences or its general physics knowledge are often irrelevant to the problem at hand. However, by including inferred or derived quantities and relations, EUREKA's "space of descriptors" for a problem changes qualitatively. This enhancement process bears a resemblance to the idea of constructive generalization [Michalski, 1983]. Constructive generalization changes the space of descriptors that defines a concept by adding descriptors that are implied by, although not explicitly contained in, any of the examples. In this respect, EUREKA is not limited to describing and learning problem types in terms of the original descriptors (the particular objects, relations, and features) that were used in the initial problem representation. As EUREKA solves more problems, it achieves a more expert-like organization of its past problem-solving experiences, and it is able to gather more potentially relevant inferences prior to generating equations.

```
Objects       body-1, surface-1
Relations     on(body-1 surface-1)
Features      moving(body-1,time2), inclined(surface-1),
              friction(surface-1), mass(body-1)
Knowns        (body-1 mass), (surface-1 incline-angle), gravity,
              coefficient-friction, (body-1 acceleration-along-y-axis)
Desired       (body-1 acceleration-along-x-axis)
Forces        (body-1 gravity-along-x-axis),
              (body-1 gravity-along-y-axis),
              (body-1 friction-along-x-axis),
              (body-1 normal-along-y-axis)
Derived       (body-1 gravity-along-x-axis),
              (body-1 sum-of-forces-along-y-axis),
              (body-1 gravity-along-y-axis),
              (body-1 normal-along-y-axis)
              (body-1 friction-along-x-axis),
              (body-1 sum-of-forces-along-x-axis),
              (body-1 acceleration-along-x-axis)
Inferences    (if a body is on a surface that has friction -->
                 then a frictional force is acting on the body)
              (if a body is on something  -->
                 then a normal force is acting on the body)
Soln-method   (TOP-GOAL (body-1 acceleration-along-x-axis))
              ((body-1 acceleration-along-x-axis) RETRIEVES Eqn-F3)
              ((body-1 acceleration-along-x-axis) HAS-SUBGOAL
                       (body-1 sum-of-forces-along-x-axis))
              ((body-1 sum-of-forces-along-x-axis) RETRIEVES Eqn-F1)
              ((body-1 sum-of-forces-along-x-axis) HAS-SUBGOAL
                       (body-1 gravity-along-x-axis))
              ((body-1 gravity-along-x-axis) RETRIEVES Eqn-F5)
              ((body-1 acceleration-along-x-axis) HAS-SUBGOAL on)
              ((body-1 acceleration-along-x-axis) HAS-SUBGOAL
                       (body-1 friction-along-x-axis))
              ((body-1 friction-along-x-axis) RETRIEVES Eqn-F8)
              ((body-1 friction-along-x-axis) HAS-SUBGOAL
                       (body-1 normal-along-y-axis))
              ((body-1 normal-along-y-axis) RETRIEVES Eqn-F2)
              ((body-1  normal-along-y-axis) HAS-SUBGOAL
                       (body-1  sum-of-forces-along-y-axis))
              ((body-1 sum-of-forces-along-y-axis) RETRIEVES Eqn-F4)
              ((body-1 normal-along-y-axis) HAS-SUBGOAL
                       (body-1 gravity-along-y-axis))
              ((body-1 gravity-along-y-axis) RETRIEVES Eqn-F6)
              ((body-1 friction-along-x-axis) RETRIEVES Eqn-F1)
```

Figure 4.2
Enhanced problem representation of Problem1

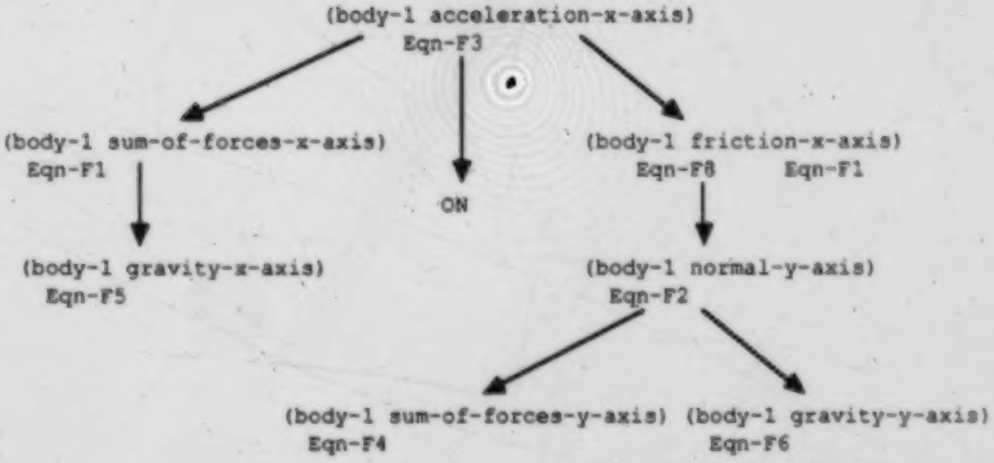


Figure 4.3
Problem1's solution method goal tree

4.3 A sample P-MOP

An example of a P-MOP is shown in Figure 4.4. This P-MOP indexes two problem representations, problem1 and problem2. Figure 4.2 shows the final problem representation for problem1. The problem statement for problem2 is as follows:

> A body of mass 3 slugs is on an frictionless inclined plane that has an angle of 40 degrees from the horizontal. Find the acceleration of the body.

The P-MOP reflects the commonalities and differences between the two problems' final representations. Norms are the common features of the problems indexed under a P-MOP. Both problems involve a moving body on an inclined surface, a gravity force, and a normal force acting on the body, several known quantities, and the acceleration of the body along the plane as the desired quantity. The following inference, used in the solution of both problems, is stored on the P-MOP:

> If a body is on an object
> Then there is a normal force acting on the body.

The inference in problem1 regarding friction is not included because problem2 does not deal with friction.

EUREKA compares the solution methods of the two problems forming this P-MOP. Common solution steps of the two methods are stored on the P-MOP. In this case, only part of the solution method was common between the two problems. The solution method shown on the P-MOP is the entire solution method of problem2. Because friction is not common between the two problems, the parts of the solution method in problem1 dealing with friction are not included in this P-MOP. This portion of the solution method dealing with friction remains on problem1's representation which is stored under the P-MOP.

In addition to containing commonalities between these two problems, the P-MOP also contains the differences by which the problems will be indexed "underneath" this P-MOP. In this example, features regarding friction are the differences. Problem2 is indexed by the feature that the surface is frictionless. Problem1 is indexed in two ways: the feature that the surface has friction and the feature that the coefficient of friction is known. Recall that indices are specific feature-value pairs: known(coefficient-friction), feature(frictionless,surface-1), and feature(friction,surface-1).

```
NORMS: Body-1 is a body
       Surface-1 is a surface
       Body-1 is moving at time2
       Body-1 is on surface-1
       Surface-1 is inclined
       Mass of body-1 is known
       Gravity is known
       Incline-angle of surface-1 is known
       Body-1 acceleration along y-axis is known
       Body-1 acceleration along x-axis is desired
       There is a Gravity force on body-1 along the x-axis
       There is a Gravity force on body-1 along the y-axis
       There is a Normal force on body-1 along the y-axis

INFERENCES:
       If a body is on something -->
             Then there is a normal force acting on the body

SOLUTION-METHOD:  (body-1 acceleration-x-axis)
                        Eqn-F3
                          ↓
                  (body-1 sum-of-forces-x-axis)
                          Eqn-F1
                          ↓
                  (body-1 gravity-x-axis)
                          Eqn-F5

NUMBER OF PROBLEMS INDEXED:  2

DIFFERENCES:  ((Feature frictionless surface-1)  problem2)
              ((Feature friction surface-1)  problem1)
              ((Known coefficient-friction)  problem1)

PREDICTIVE FEATURES:

NON-PREDICTIVE FEATURES:
```

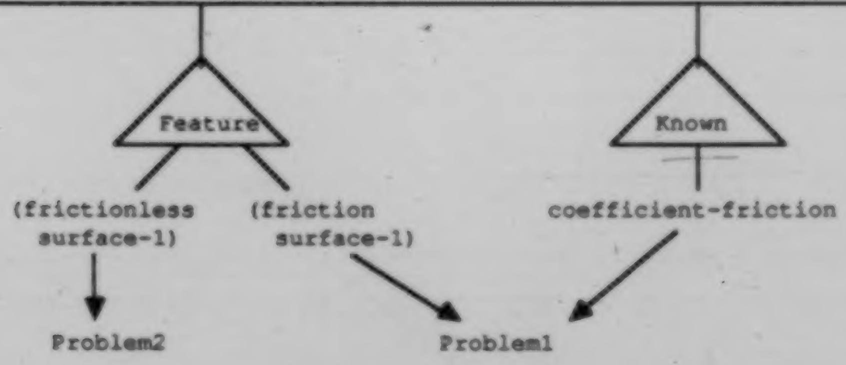


Figure 4.4 A sample P-MOP

The last type of information carried on P-MOP's are lists of predictive and non-predictive features. As more problems are indexed under this P-MOP, features that make good indices are recorded on the predictive list and features that do not make good indices are recorded on the non-predictive list. Because non-predictive features are never used to traverse the P-MOP network, the stored problem representations and P-MOPs that these non-predictive differences lead to are functionally forgotton.

There is an important difference between Kolodner's E-MOP system and EUREKA's P-MOP system. Kolodner's E-MOPs track commonalities and differences between the features that describe the events indexed by the E-MOP. In contrast, EUREKA actively solves problems and the type of knowledge it is acquiring during solution is more complex than features that occur in problem descriptions. A problem representation includes all features in the original specification of the problem as well as all derivations, inferences, and solution methods added to the problem to obtain a solution. To acquire expert solution methods, EUREKA keeps track of commonalities between the solution methods and inferences used in the problems it solves. Common inferences and portions of solution methods move around in the P-MOP network until they characterize the correct set of problems. The changes to the mechanisms required to store the richer representation will be described in section 4.4.3.

4.4 The EUREKA Model

4.4.1 The Semantic Network

The semantic network holds EUREKA's unorganized knowledge and reflects the minimal knowledge a novice might have about Newton's force laws and conservation of energy. The assumption is that this textbook physics knowledge is accessible to the novice, but in a relatively unorganized manner. The basic physics knowledge included in the network are concepts, inferences, and equations. Figure 4.5 shows examples of the knowledge contained in the semantic network.

BODY	A-KIND-OF	Object
	IN-INFERENCES	Inf-A, Inf-B, Inf-C, Inf-D, Inf-E, Inf-F, Inf-G, Inf-H, Inf-I
SURFACE	A-KIND-OF	Object
	IN-INFERENCES	Inf-A
ON	A-KIND-OF	Relationship
	IN-INFERENCES	Inf-A, Inf-B, Inf-D, Inf-F, Inf-H
MASS	A-KIND-OF	Feature
	FEATURE-OF	Object
	IN-EQUATIONS	Eqn-F3, Eqn-F4, Eqn-F5, Eqn-F6, Eqn-F7, Eqn-E4, Eqn-E5, Eqn-E6, Eqn-E7, Eqn-E8, Eqn-E9
ACCELERATION	INCLUDES	Accleration-X-Axis, Accleration-Y-Axis
	A-KIND-OF	Concept
	IN-EQUATIONS	Eqn-F3, Eqn-F4, Eqn-K1, Eqn-K2, Eqn-K3, Eqn-K4, Eqn-K5, Eqn-K6
FORCE	INCLUDES	Sum-Of-Forces-X-Axis, Sum-Of-Forces-Y-Axis
	A-KIND-OF	Concept
	IN-EQUATIONS	Eqn-F1, Eqn-F2, Eqn-F3, Eqn-F4
GRAVITY	INCLUDES	Gravity-X-Axis, Gravity-Y-Axis
	A-KIND-OF	Concept
	IN-EQUATIONS	Eqn-F1, Eqn-F2, Eqn-F5, Eqn-F6, Eqn-F7, Eqn-E8, Eqn-E9
EQN-F1	A-KIND-OF	Equation
	LHS	Sum-Of-Forces-X-Axis
	RHS	
	PROCEDURE	Sum all of the forces in the problem for the RHS of the equation
EQN-F3	A-KIND-OF	Equation
	LHS	Sum-Of-Forces-X-Axis
	RHS	Mass, Accleration-X-Axis
EQN-F5	A-KIND-OF	Equation
	LHS	Gravity-X-Axis
	RHS	Mass, Gravity, Incline-Angle
	CONSTRAINT	Body must be on surface
INF-A	A-KIND-OF	Inference
	CONDITION	If a body is on a surface And the surface has friction involved
	ACTION	Then there is a friction force acting on the body
INF-B	A-KIND-OF	Inference
	CONDITION	If a body is on an object
	ACTION	Then there is a normal force acting on the body

Figure 4.5 Semantic Network

Concepts in the semantic network include physical objects, features of objects, and relationships between objects. An example of a physical object is a *body*. A body can have "mass" as a feature and the relationship "on" with a surface. Physics concepts such as forces and energies are also included in the network. These include gravity, friction, and normal forces as well as kinetic and potential energy.

Inferences are rules that describe a group of features in their left-hand side necessary to infer the conclusion on the right-hand side. For instance, INF-B states that "If there is a body on some object then I can infer the presence of a normal force acting on the body." The two conditions necessary for inferring friction, INF-A, are that a body must be on a surface and the surface must have a friction component.

The third type of knowledge contained in EUREKA's semantic network are the physics equations. These equations describe kinematic, force, and energy relationships. The equations contain a left-hand side, a right-hand side, and constraints. Constraints of the equation must be fulfilled in order for the equation to be used. The constraint of Eqn-F5 in Figure 4.5 requires the object of interest must be on a surface. This must appear in the problem representation in order for the equation to be used.

There are two force equations that are problematic to represent. These two equations sum all the relevant forces acting on a body along the x-axis and the y-axis. The relevant forces acting on a body are not known a priori because these forces vary from problem to problem. Because these two equations have a right-hand side that include variables that are relevant depending on the situation, they contain a procedure that specifies how the right-hand side is to be constructed. Eqn-F1 is such an equation: a procedure specifies that the right-hand side is to be constructed by summing all of the forces acting along the x-axis. For example, if a block were on a frictionless, inclined plane, only gravity-x-axis would appear on the right-hand side. If there were a friction component, then both gravity-x-axis and friction-x-axis would appear on the right-hand side.

When EUREKA works with these two equations, it does not know whether all of the relevant forces have been identifed and added to the right-hand side of these equations. For this reason, when EUREKA thinks it has solved these types of equations (i.e., all but one variable in the equation are known), it asks the user to confirm that all of the force components are included and the equation is indeed solved. If all of the force components are present in the right-hand side of Eqn-F1, the user replies "yes", the equation is marked as solved, and the unknown variable is marked as derived. If force components are missing, the user responds "no" and the equation remains unsolved.

Concepts are associated with the inferences and equations in which they participate. A concept is associated with an inference if the concept appears in the condition side of the inference. As shown in Figure 4.5, the concept "on" appears in the condition side of five inferences. A concept is associated with an equation if it appears anywhere in the equation. In Figure 4.5, acceleration is a concept appearing as a variable in eight equations. The next section describes how the information contained in a problem representation "drives" the retrieval of inferences and equations from the semantic network during problem solving.

4.4.2 Means-ends analysis

A means-ends analysis problem solver, similar to the one in Larkin's [1981] model, is used to solve problems. The means-ends problem solver works backwards from the desired quantity, retrieving equations and using the unknowns of the retrieved equation to retrieve more equations. Unlike Larkin's model, EUREKA also retrieves and uses inferences to assist in problem solving. Inferences are retrieved for every concept considered. If EUREKA uses up all of its equations for a concept and has not solved the problem, it shifts its attention to other features of the problem that may have resulted from inferences. These features in turn might enable EUREKA to continue its equation-retrieval process. As the means-ends problem solver works on the problem, all the information retrieved from the semantic network to solve the problem is added to the problem representation. The solution method that evolves during this process is also added to the problem representation. A problem is solved when its desired quantity has been derived.

Figure 4.6a and Figure 4.6b show the general control scheme for the means-ends analysis problem solver. The most important decisions concern selecting a "focus of attention", retrieving associated inferences and equations, and setting up goal and subgoal structures. The circles in Figure 4.6b indicate when EUREKA asks the user for input. One circle asks if the equation is solved. Recall that this question is only asked for two equations, the equations that sum the forces along the x and y axis. The other circle asks the user if the problem is solved. Because EUREKA does not manipulate equations algebraically, it is difficult for it to detect a solution when simultaneous equations are involved and therefore EUREKA will ask the user to confirm or disconfirm that a solution has been reached.

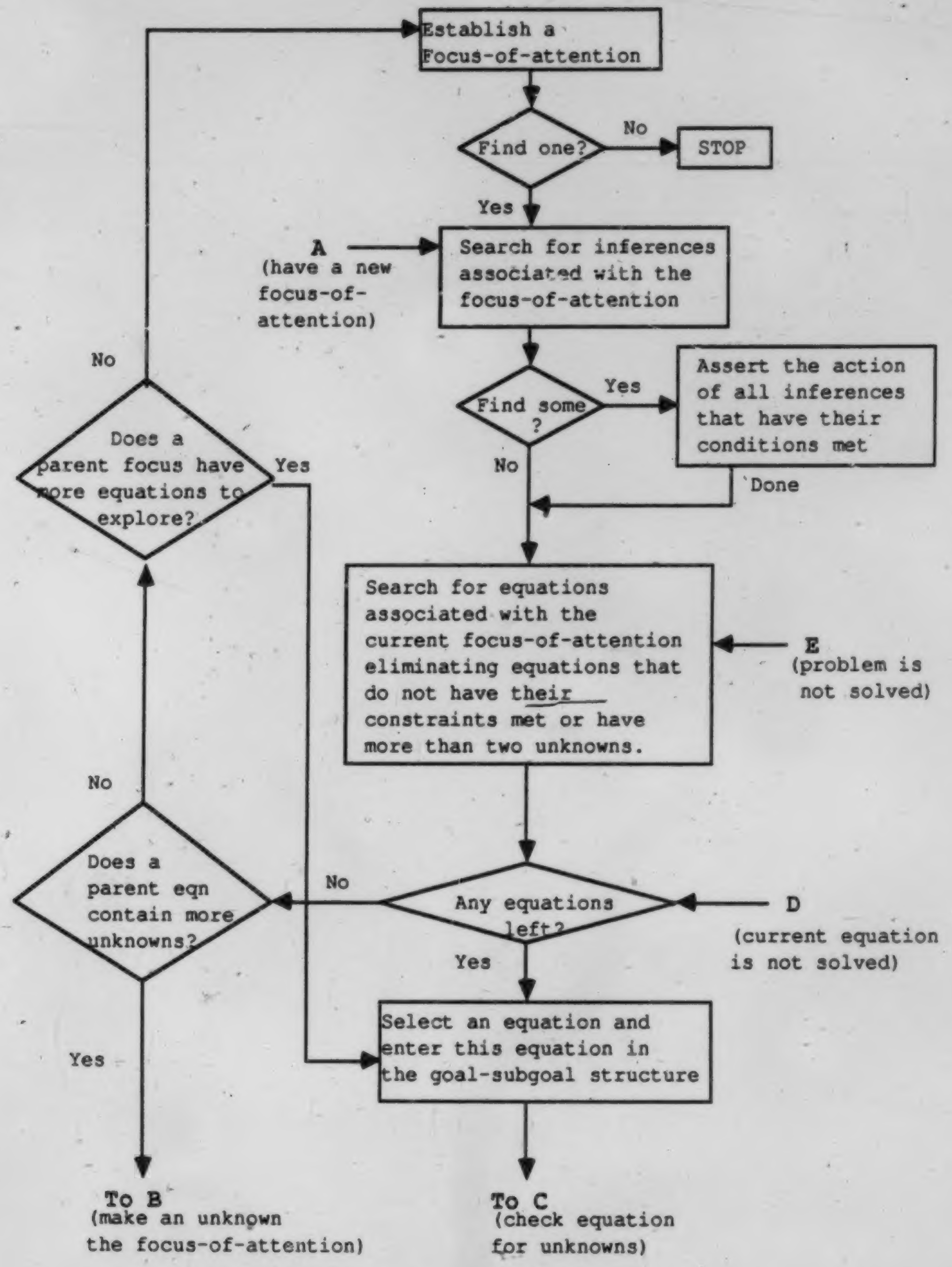

Establish a Focus-of-attention
Find one?
No
STOP
Yes
A
(have a new focus-of-attention)
Search for inferences associated with the focus-of-attention
Find some ?
Yes
Assert the action of all inferences that have their conditions met
No
Done
No
Does a parent focus have more equations to explore?
Yes
Search for equations associated with the current focus-of-attention eliminating equations that do not have their constraints met or have more than two unknowns.
E
(problem is not solved)
No
Does a parent eqn contain more unknowns?
No
Any equations left?
D
(current equation is not solved)
Yes
Select an equation and enter this equation in the goal-subgoal structure
Yes
To B
(make an unknown the focus-of-attention)
To C
(check equation for unknowns)

Figure 4.6a
Control structure for problem solver
(Continued on next page)

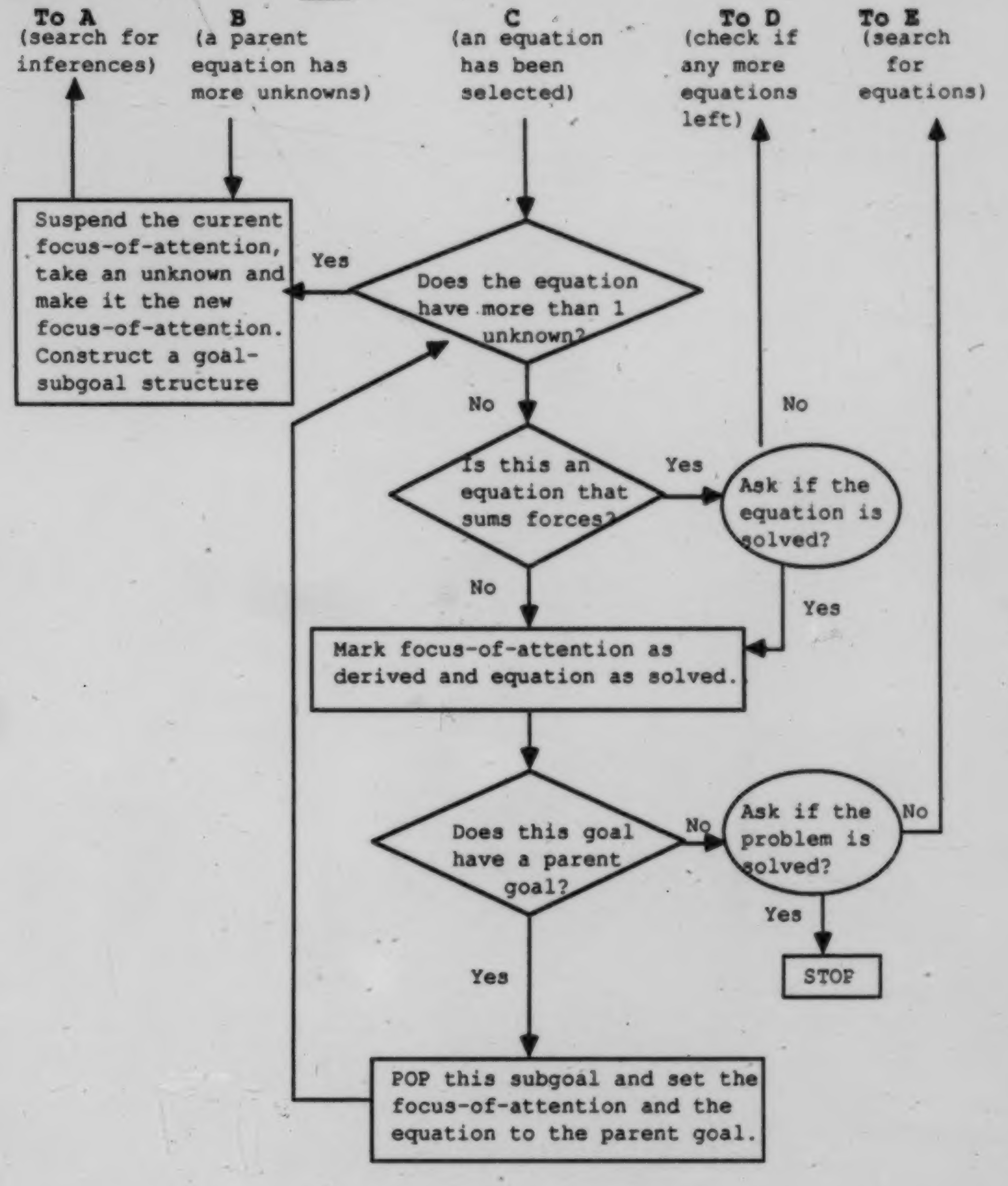

To A
(search for inferences)
B
(a parent equation has more unknowns)
C
(an equation has been selected)
To D
(check if any more equations left)
To E
(search for equations)
Suspend the current focus-of-attention, take an unknown and make it the new focus-of-attention. Construct a goal-subgoal structure
Yes
Does the equation have more than 1 unknown?
No
No
Is this an equation that sums forces?
Yes
Ask if the equation is solved?
No
Yes
Mark focus-of-attention as derived and equation as solved.
Does this goal have a parent goal?
No
Ask if the problem is solved?
No
Yes
STOP
Yes
POP this subgoal and set the focus-of-attention and the equation to the parent goal.

Figure 4.6b
Control structure for problem solver

I will step through an example to demonstrate how EUREKA works. The input to EUREKA is the original problem representation given in Figure 4.1. The final enhanced problem representation was given in Figure 4.2. The part of the semantic network that was useful during the solution was shown in Figure 4.5.

The first focus-of-attention for the means-ends problem solver is always the desired quantity of the problem. In this case, it is acceleration of the body along the x-axis. This is set as a goal and added to the problem representation. EUREKA first retrieves all inferences from the semantic network associated with the current focus-of-attention. In this case, there are none associated with acceleration. EUREKA then retrieves from the network all equations that have the focus-of-attention as a variable. After discarding equations that do not have their constraints met or have more than two unknown variables, EUREKA is left with Eqn-F3:

sum-of-forces-x-axis = mass * acceleration-x-axis.

Having retrieved this equation, EUREKA works on deriving the remaining unknowns so the equation can be solved for acceleration. The mass of the body appears as a known quantity in the problem representation, so the next focus-of-attention is sum-of-forces-x-axis. The solution method of the problem representation shown in Figure 4.2 is evolving. At this stage of the problem, the solution method records that (a) the first focus-of-attention was acceleration-along-x-axis, (b) acceleration-along-x-axis retrieved Eqn-F3, and (c) the next focus-of-attention was sum-of-forces-x-axis.

The focus-of-attention is now sum-of-forces-x-axis. EUREKA finds no inferences associated with this concept but does retrieve four equations from the semantic net. Only equation Eqn-F1 has all its constraints satisfied and does not have more than two unknown variables. As mentioned previously, this equation's right-hand side is constructed by a procedure which sums all of the forces along the x-axis. The only force along the x-axis that is described by the problem at this point in time (see Figure 4.1) is gravity. Therefore, the equation becomes

Sum-of-forces-x-axis = Gravity-x-axis

and the next focus-of-attention is gravity-x-axis.

Gravity is not associated with any inferences but does appear in equation Eqn-F5. All variables in this equation have a status of known. Therefore gravity-x-axis is marked as derived[2] and added to the problem representation. This goal is "popped", EUREKA goes back to Eqn-F1, and the focus-of-attention reverts to sum-of-forces-along-x-axis. As it is currently constructed, Eqn-F1 has only one unknown, sum-of-forces-x-axis. Because Eqn-F1 sums forces, EUREKA asks the user to confirm that Eqn-F1 is solved. The force due to friction is missing, so the user responds "no" and the equation remains unsolved. There are no other equations associated with the current focus-of-attention, sum-of-forces-along-x-axis. Therefore, EUREKA changes its focus-of-attention back to the parent goal, acceleration-along-x-axis. However, there are no other eligible equations for deriving acceleration.

Once EUREKA has arrived back at the top goal with no answer and no more equations to use, it must find something else to focus on. There are a number of other aspects about the problem that could serve as a focus-of-attention and possibly get the solution moving again. Specifically, there are four leads to follow: things recently inferred (new concepts not originally in the problem representation), relations among objects in the problem, features of objects in the problem, and then objects themselves. EUREKA considers these four leads in that order. Intuitively, it seems reasonable that the most recent changes to the representation would influence the solution method. In some ways, this keeps EUREKA on the same tangent it originally began, even though it has run out of equations. This assumption, as well as the ordering of relations, then object features, and finally objects, as focus-of-attention candidates, may not be a correct characterization of novice behaviour. The implications of this are discussed later in the thesis.

---

[2] The system does not manipulate equations algebraically. When all but one of the variables in an equation have a status of known or derived, the remaining variable achieves a derived status.

In the example being followed, there are no "new" concepts yet, therefore EUREKA makes the relation "on" its next focus-of-attention. EUREKA retrieves five inferences concerning "on" from the network. If all conditions of the left-hand side of the inference appear as features in the problem representation, both the inference rule and the result of its action are added to the problem representation. Two inferences have their conditions met:

Inf-A If a body is on a surface
and the surface has friction involved
Then there is a friction force acting on the body

Inf-B If a body is on an object
Then there is a normal force acting on the body

The problem representation is augmented with the information that the body has a frictional force acting on it along the x-axis and a normal force acting along the y-axis. EUREKA goes to the semantic network to retrieve equations but the concept "on" is not associated with any equations. Because this is a dead-end another focus of attention is required. However, the problem representation now has two new concepts, so EUREKA establishes one of them (in this case, friction-x-axis) as the new focus-of-attention and goes on to retrieve associated inferences and equations. EUREKA continues to retrieve and solve equations, deriving quantities until the original desired quantity is derived and the problem is solved. The problem representation is enhanced with any inference rules that were used as well as the solution method. When the problem is solved, its final problem representation is stored in the P-MOP network.

4.4.3 Storing the enhanced problem representation

The P-MOP network represents the organized information of the model. EUREKA's problem-solving experiences, recorded on the final problem representation, are stored in the P-MOP network. In order to manage the richer knowledge that is involved with problem solving (inferences and goal-subgoal structures), several changes were made to Kolodner's [1984] mechanisms. This section describes the initial P-MOP network as well as the changes made to Kolodner's mechanisms.

EUREKA starts with a "generic physics problem" P-MOP as its initial structure. The norms contained on this generic physics problem P-MOP are as follows:

physics problems have knowns
physics problems have a desired quantity
physics problems have objects
objects have features
objects can have relationships with other objects.

No other information is carried in this initial P-MOP, which serves as the root node of the network. Problem representations can be indexed in the P-MOP network under the objects, relations, features, knowns, and desireds of the problem. Features that originally describe the problem as well as any additional features added to the problem are potential indices to traverse and index problems in the P-MOP network. In this way, abstract physics concepts such as normal force, inferred from groups of features via some inference rule, will eventually be eligible to become indices or norms.

When a problem representation traverses the P-MOP network, it can arrive at either a P-MOP or another problem representation. Recall that to arrive at a P-MOP or stored problem representation, the incoming problem must have followed one or more indices. As noted earlier, this is similar to passing a series of tests. I will first consider the case of a new enhanced problem representation arriving at a P-MOP. The old P-MOP is reconsidered in light of this new problem-solving experience with which it shares some commonalities. At this point, the existing P-MOP may be refined. Any of the P-MOPs norms that do not appear in the problem representation are "disconfirmed" and removed from the P-MOP. Unlike Kolodner's system, which disconfirms a norm only when its certainty falls below a pre-specified threshold, EUREKA disconfirms a norm the first time it does not appear in a problem representation indexed by the P-MOP. In this way, it is certain that the norms of a P-MOP are true of a great majority of the problems it organizes.

Disconfirmed norms are, of course, differences between a P-MOP and the incoming problem and they will now be indices to a new organization "beneath" this P-MOP. When disconfirmed norms are taken off the current P-MOP and moved down to new sub P-MOP's, any inferences in which they appeared go with them. Also, any parts of the solution method residing on a P-MOP that does not also appear in the incoming problem representation are moved down to the sub P-MOPs. As a result, partial solution methods can reside on P-MOPs.

Figure 4.7 schematically shows a before-and-after picture of a P-MOP that is checked against an incoming problem representation. The features of the problem representation are compared against the norms of the P-MOP7. Because the norms A and C do not appear in the problem representation, they are removed from P-MOP7 and become indices to a new P-MOP, P-MOP22. P-MOP22 is indexed in two ways. The inference in which the disconfirmed norms appear are also moved down to P-MOP22. The solution method of the problem representation is compared to that of P-MOP7. The solution step, C (Eqn-F8), that appears in P-MOP7 and not the problem representation, moves down to P-MOP9 and P-MOP22. A partial solution method now sits on P-MOP7 and if the extra solution step, C (Eqn-F8), is required, an index (difference) must be travelled to arrive at a P-MOP one level further down in the network.

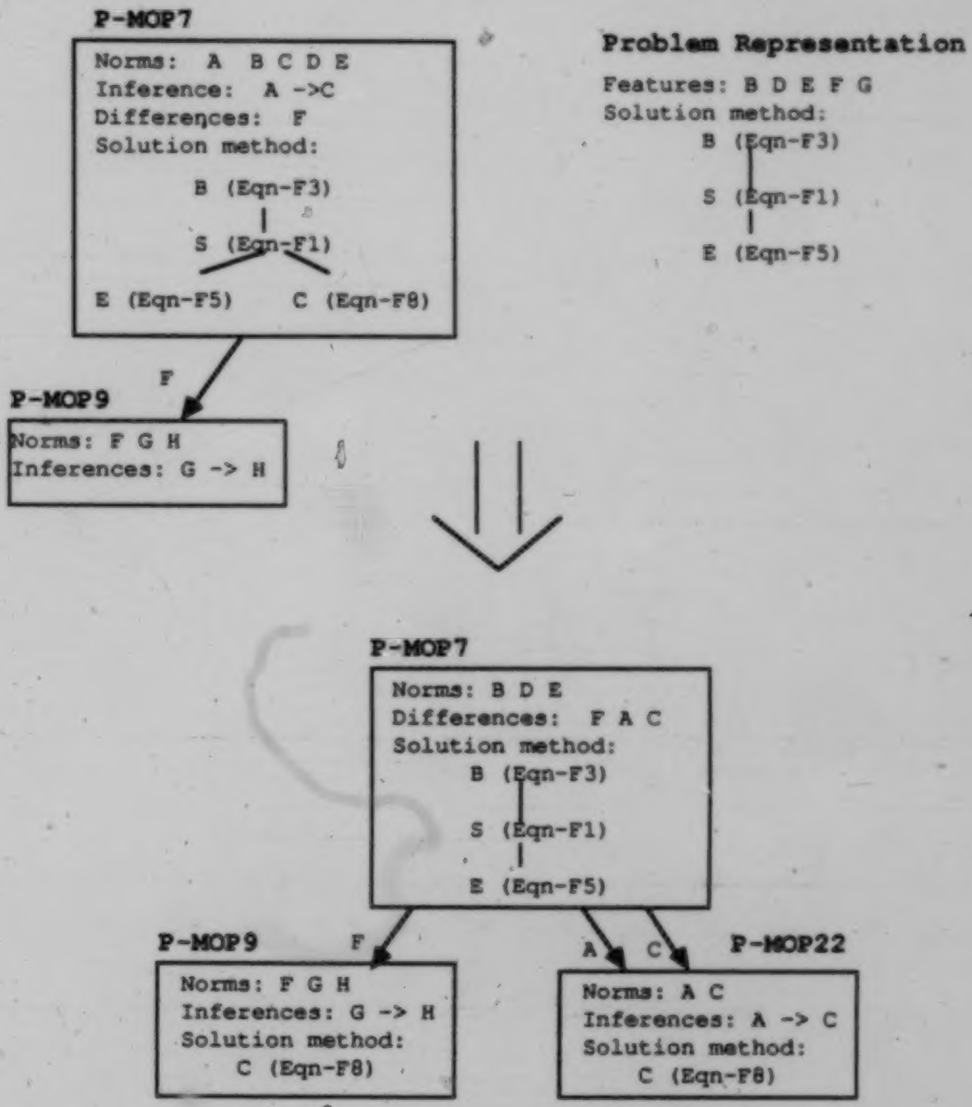


Figure 4.7
Checking the information on a P-MOP against an incoming problem representation

An incoming problem representation can also arrive at a stored problem in the network. As in Kolodner's model, a new P-MOP is created. This new P-MOP is indexed by the feature that previously indexed the stored problem. All features in common between the two problems are made into norms on this new P-MOPs. In addition, all inferences and solution steps common between the two problems are also stored on the P-MOP. The two problems are then indexed under the P-MOP according to their differences. Figure 4.8 shows the new P-MOP that is created as a result of an incoming problem representation colliding with a stored problem. Feature F that indexed problem1 now indexes P-MOP12. The features, inferences, and solution steps in common between problem1 and problem2 are stored on P-MOP12. Problem1 is indexed under P-MOP12 by features G and H and problem2 is indexed by feature E. These features are the differences between problem1 and problem2.[3]

When a sub-MOP indexes a majority of its parent's problems (i.e., the ratio of problems that the sub-MOP indexes exceeds a predefined threshold), this means that the current organization is not the best. Specifically, a feature that is acting as a difference is really a commonality. As a result, the sub-MOP that is indexed by this difference collapses into its parent P-MOP. All of the information contained on the sub-MOP is moved to the parent P-MOP and the sub-MOP is destroyed. This is illustrated schematically in Figure 4.9. After indexing a number of problems, P-MOP12 organized a majority of its parent P-MOP's problems. This means that feature F was common to most of the problems indexed under P-MOP7. As a result, feature F becomes a norm on P-MOP7 and P-MOP12 collapses into P-MOP7. All of P-MOP12's norms, inferences, and solution steps are moved up to P-MOP7 and P-MOP12 and its substructure is destroyed. Note that the indices J, K, and L that lead to P-MOPs and stored problems are still valid differences for indexing. Also, problem1 and problem2 are not "lost" because they are likely to be indexed elsewhere in the P-MOP network (this is demonstrated in the trace of Chapter 5).

---

[3] Currently all of the information associated with a problem is stored in the P-MOP structure. A more storage effective method would only store what was *not* mentioned by the structure above the problem.

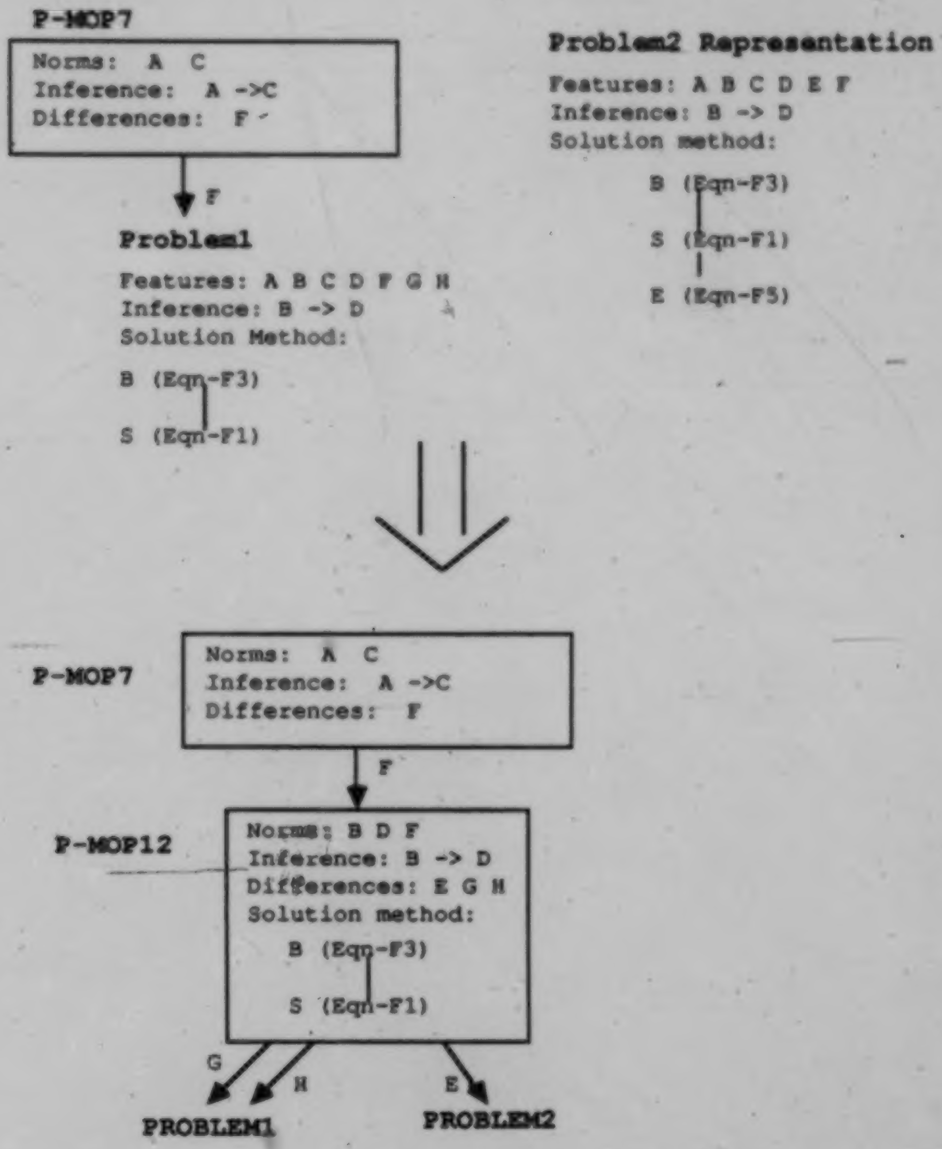


Figure 4.8
Arriving at a stored problem and creating a new P-MOP

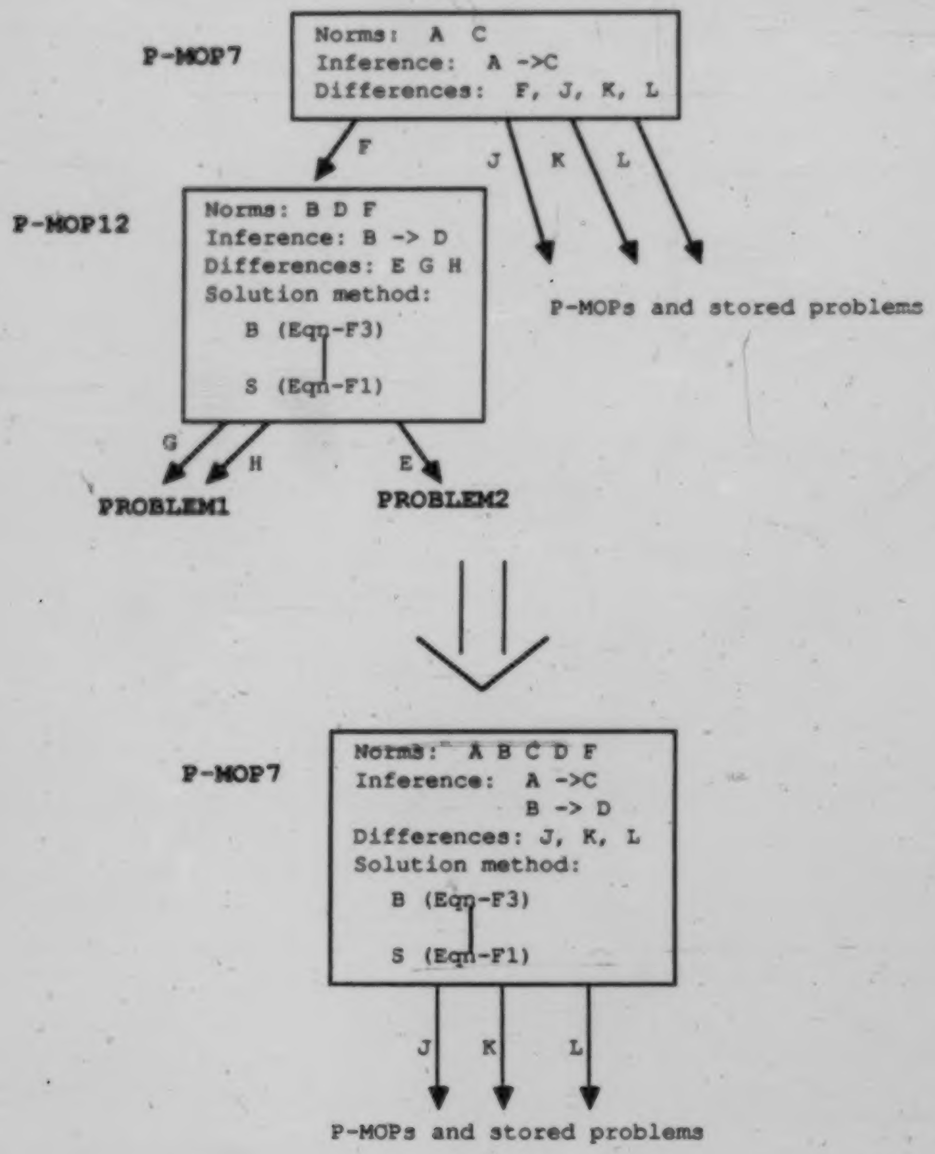


Figure 4.9
Collapsing a P-MOP

4.4.4 Retrieving past problem solving knowledge from P-MOPs

The previous sections have described how EUREKA uses a means-ends method to solve a problem, how a problem representation is enhanced with the inferences and methods used during its solution, and how the final enhanced problem representation is indexed into the P-MOP network. Unless this P-MOP knowledge is consulted during problem solving, it will, of course, have no impact on problem-solving behaviour. When given a problem, EUREKA examines the organized knowledge in the P-MOP network to assist in the solution. EUREKA identifies P-MOPs that are relevant to its current problem, but the current implementation makes only limited use of the information it finds. Ideally, it should execute the solution methodologies it finds in the P-MOP network.

I will now outline what is currently implemented for using the P-MOP knowledge. The features contained in the initial representation of the problem are used as indices to traverse the P-MOP network. All of the information contained in the problem representation of Figure 4.1 are potential indices into the P-MOP network. If a feature of the problem appears as a difference in a P-MOP, the corresponding index is traversed. When faced with a group of indices to traverse, EUREKA will first prefer the index leading to the P-MOP that organizes the *most* problem solving experiences. Thus, indices are traversed in an order determined by the number of problems they index. This is important because EUREKA is directed to patterns matched most often in the past. In this way, EUREKA is sensitive to frequency, which is an important feature of most models of human learning.

As the P-MOP network is traversed, inferences found on P-MOPs or stored problems may be relevant to the current problem. If the current problem contains the condition features of the inference in its representation, then the conclusion of the inference and the inference itself is added to the enhanced problem representation. Finding relevant inferences in the P-MOP network will influence the solution constructed with means-ends analysis. If EUREKA has not solved many problems of the type it is currently trying to solve or has not solved many problems at all, traversing the P-MOPs is going to yield little useful information for the current problem. After traversing the P-MOP network EUREKA uses means-ends analysis to complete the solution.

Although EUREKA finds solution methods during its traversal of the P-MOP network, the procedures for utilizing them are currently not implemented. The reason EUREKA does not use solution methods it finds in the P-MOP network is related to the representation of these solution methods and will be discussed later in the thesis. Procedures for utilizing the solution method would enable EUREKA to reconstruct solutions found in the P-MOPs or stored problems. For example, recall that the P-MOP shown in Figure 4.4 holds a solution method that was common to the two problems this P-MOP indexes. An incoming problem with a desired quantity of acceleration-x-axis would be able to use this solution method. If the incoming problem did not involve friction, the solution method stored on this P-MOP would be the entire solution method of the problem. However, if the incoming problem had a friction component then the solution method stored on the P-MOP would be a partial solution. The incoming problem would then traverse the indices feature(friction surface-1) and known(coefficient-friction) and find problem1. Recall that problem1 contains the additional solution steps needed for frictional forces. In the complete implementation, enough inferences and solution methods could be gathered from the P-MOP network to solve this problem completely in this forward working fashion, without resorting to means-ends analysis.

Chapter 5

Trace

This chapter presents a trace of EUREKA as it solves physics problems and stores its problem-solving experiences. The purpose of the trace is to demonstrate the evolution of EUREKA's knowledge from a novice-like organization to a more expert-like organization. At the end of the trace, a schematic view of the resulting P-MOP network will be related to expert knowledge structures described by Chi et al. [1981].

EUREKA solved ten physics problems. These problems fall into four categories. The categories include: inclined plane problems solvable with a force method, inclined plane problems solvable with an energy method, pulley problems solvable with a force method, and pulley problems solvable with an energy method. These problems have two different "surface stories", inclined planes and pulleys, and two different solution principles, force methods and energy methods. From the empirical evidence [Chi et al., 1981], one would expect a novice to categorize these problems by their surface features, and an expert to categorize these problems by their solution principles. Initially, EUREKA organizes its experiences according to the surface features of the problems. As EUREKA solves more problems, commonalities among solution methods, associated physics concepts, and inferences emerge via the P-MOP mechanisms. EUREKA's P-MOP knowledge base develops into a more expert-like organization as problem-solving experiences are organized around major physics principles and solution methodologies. This organization, in turn, influences the basic approach to problem solving.

EUREKA starts with the generic physics P-MOP that serves as the "root node" of the network that will evolve. EUREKA knows that problems have known quantities, desired quantities, and objects, and that objects have features, and relationships with other objects. The number in the upper right-hand corner indicates that this P-MOP does not store any experiences.

```
 MOP1                                          0

Norms: Problems have knowns
       Problems have desireds
       Problems have objects
       Objects have features
       Objects have relationships with other objects
```

Before applying means-ends analysis to a new problem, EUREKA uses the features of the problem to traverse the P-MOP network and retrieve information to assist in the solution. As mentioned in the previous chapter, the entire model is not implemented - EUREKA currently uses only inferences found in the P-MOP network. This information alone affects the solution approach and is demonstrated in the trace. Because the current implementation does not use the solution methods (the goal-subgoal trees with equations) found in the P-MOP network, all problems are solved using means-ends analysis. This does not affect the overall organization of the P-MOP network, but only the *types of solutions* stored in the problem solving experiences and subsequently on the P-MOPs. The implications of this for evaluating the general framework are, I believe, negligible. This will be discussed further in the general discussion in Chapter 6.

The remainder of the chapter will detail a trace of EUREKA solving ten physics problems. Each section of the trace contains the problem specification, information on how EUREKA solves the problem, and the effect of indexing the problem solving experience in the P-MOP network. Diagrams of the developing P-MOP network illustrate the evolution of EUREKA's P-MOP knowledge base. These diagrams contain abstracted information to avoid overburdening the reader with detail. P-MOPs are described in a few words that characterize the nature of their norms. For example, a P-MOP that contains inclined-plane features as norms will be described as an Inclined-Plane Schema, whereas a P-MOP that contains energy features such as initial-kinetic-energy and initial-potential-energy are known will be described as an Energy Schema. A P-MOP with only one norm contains no description. It is important to recall that knowledge about solving energy and force problems is really distributed across several P-MOPs. That is, a single P-MOP does not correspond to a *complete* schema in the usual sense. As the P-MOP network gets large, indices leading to P-MOPs will also be described by a few words, such as "pulley indices."

Problem 1:

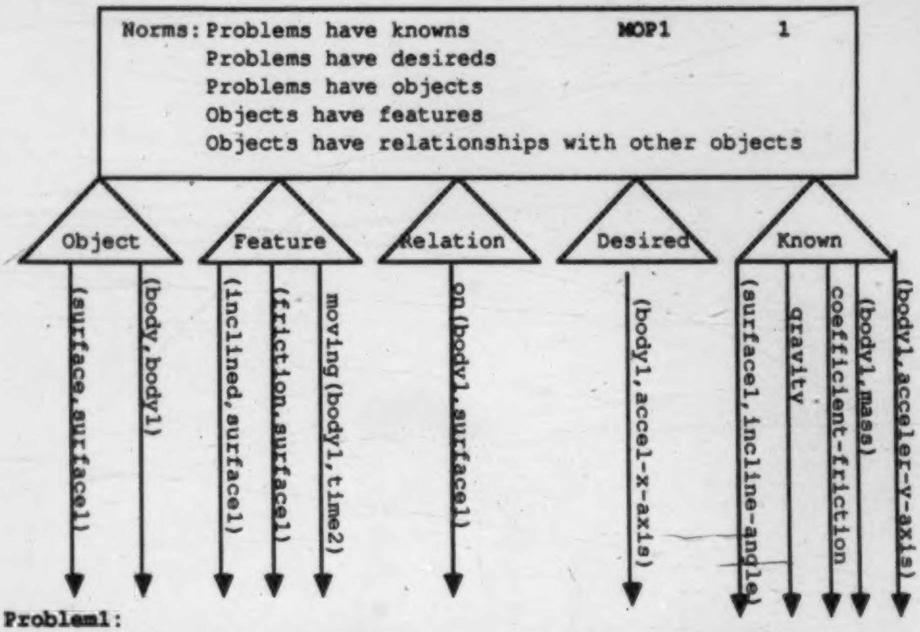
A body of mass 2 slugs is on an inclined plane that has an angle of 30 degrees from the horizontal. The coefficient of friction for the plane is 0.3. Find the acceleration of the body.

Solution:

There are no problem solving experiences stored in the P-MOP network to assist in the solution of problem1. The problem is solved using means-ends analysis.

P-MOP Reorganization:

The full enhanced representation of Problem1 is indexed under MOP1 by a number of its features. The resulting P-MOP network is shown in Figure 5.1. Indices are represented by triangles and index properties appear on lines leading to problem1. Thus, there is an index, feature = inclined surface1, and it points to the enhanced representation of problem1.



Problem1:

Features: known(body1,acceleration-y-axis) known(body1,mass) known(gravity)
known(coefficient-friction) known(surface1,incline-angle)
desired(body1,acceleration-x-axis) relation(on(body1,surface1))
feature(moving(body1,time2)) feature(friction,surface1)
feature(inclined,surface1) object(body,body1) object(surface,surface1)
force(body1,gravity-x-axis) force(body1,gravity-y-axis)
force(body1,friction-x-axis) force(body1,normal-y-axis)

Derived: (body1,gravity-x-axis) (body1,sum-forces-y-axis)
(body1,gravity-x-axis) (body1,normal-y-axis) (body1,friction-x-axis)
(body1,sum-forces-x-axis) (body1,acceleration-x-axis)

Inferences:
If a body is on a surface and the surface has friction involved
Then there is a friction force acting on the body
If a body is on an object
Then there is a normal force acting on the body

Solution:

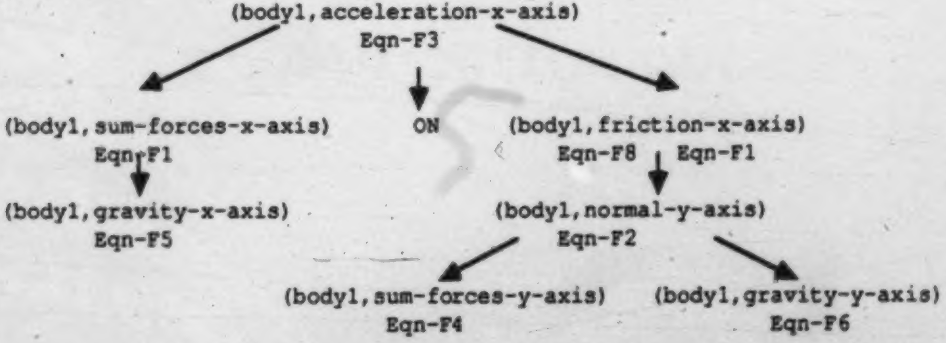


Figure 5.1 P-MOP network after Problem1 is indexed

Problem 2:

A body of mass 3 slugs is on a frictionless inclined plane that has an angle of 40 degrees from the horizontal. Find the acceleration of the body.

Solution:

The P-MOP network of Figure 5.1 is traversed, using features of problem2. Problem1 provides the following two inferences:

```
If a body is on a surface and the surface has friction involved
Then there is a friction force acting on the body.

If a body is on an object
Then there is a normal force acting on the body.
```

The first inference is not used because the inclined plane in this problem is frictionless. The second inference is not necessary for the solution of this problem, but it is added to the enhanced problem representation because its condition is true for the problem. In other words, EUREKA is at this stage adding irrelevant inferences to its enhanced problem representation. The problem is solved using means-ends analysis.

P-MOP Reorganization:

A second P-MOP has developed, MOP2. This MOP reflects the similarities and differences between problem1 and problem2. Both problems involve inclined planes, therefore MOP2 contains incline plane features as norms and will be referred to as an Inclined-Plane Schema. The common solution steps of both problems are abstracted as a sub-method and stored as Method-A on MOP2. MOP2 contains the following information:

```
Norms: known(body1,mass)   known(gravity)
       known(surface1,incline-angle)
       known(body1,acceleration-y-axis)
       desired(body1,acceleration-x-axis)
       feature(moving(body1,time2)  feature(inclined,surface1)
       relation(on(body1,surface1))  object(body,body1)
       object(surface,surface1)  force(body1,gravity-x-axis)
       force(body1,gravity-y-axis)  force(body1,normal-y-axis)
Inferences:
       If a body is on an object
       Then there is a normal force acting on the body.
Solution Method:  Method-A (Force)
                  (body1,acceleration-x-axis)
                  Eqn-F3    ↓
                  (body1,sum-of-forces-x-axis)
                  Eqn-F1    ↓
                  (body1,gravity-x-axis)
                  Eqn-F5
```

The problems are indexed under MOP2 according to their differences.

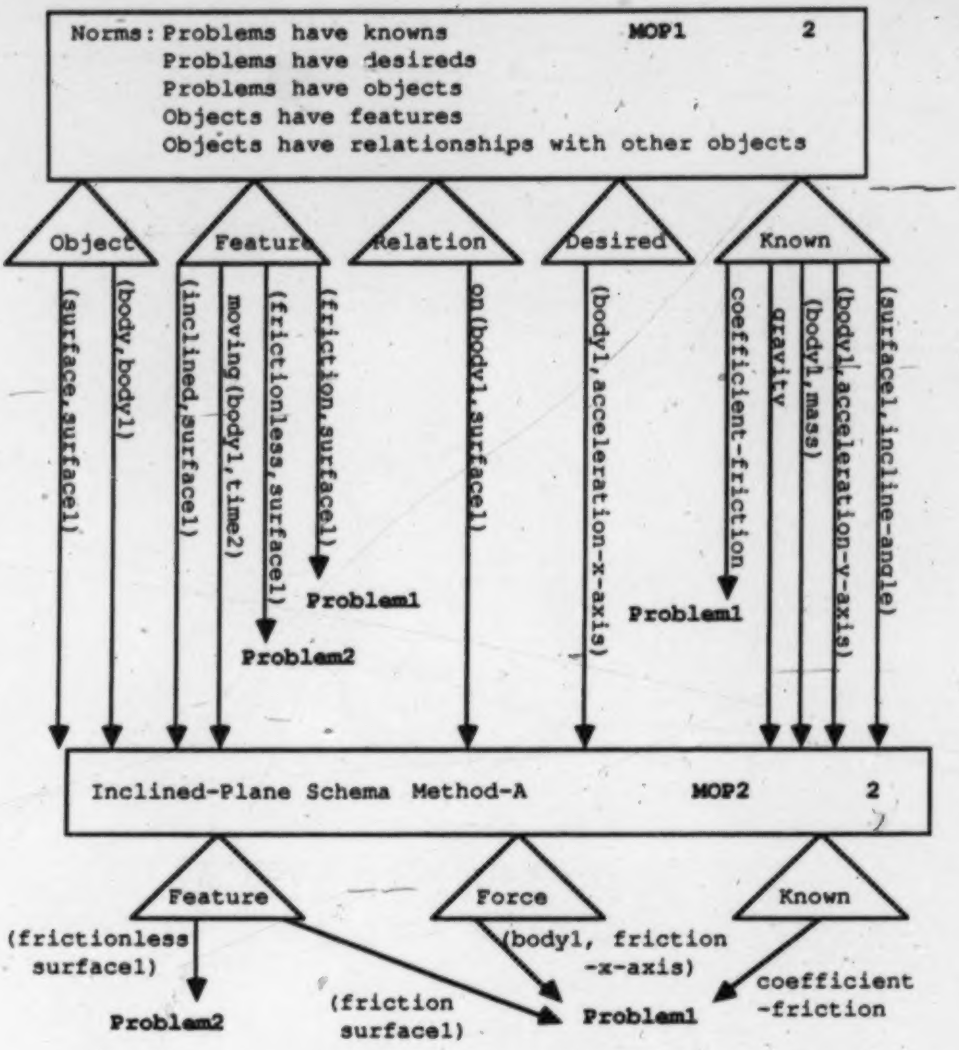


Figure 5.2 P-MOP network after Problem2 is indexed

Problem 3:

Consider two unequal masses connected by a rope which passes over a frictionless and massless pulley. The mass of body1 is 1 slug and the mass of body2 is 2 slugs. Find the acceleration of body1.

Solution:

When problem3 traverses the P-MOP network of Figure 5.2, MOP2 is reached by four indices, object = (body,body1), feature = (moving(body1,time2)), known = (gravity) and known = (body1,mass). The inference residing on MOP2 is of no value to problem3 because its conditions match patterns found in inclined plane problems and these patterns do not occur in this particular pulley problem. The problem is solved using means-ends analysis.

P-MOP Reorganization:

Two new MOPs are created, MOP3 and MOP4. The resulting P-MOP network is shown in Figure 5.3. Four of the indices leading to MOP2 were features of problem3, a pulley problem. Comparision of the pulley problem to MOP2 led to the reorganization of MOP2's information onto MOP3 and MOP4. MOP3 contains the five features common to all three problems seen thus far. MOP3 does not contain any inferences or solution method because there were none common to all three problems. MOP4 contains the features, inference and solution method that were disconfirmed by the current pulley problem from the four instances of MOP2 that this problem traversed. MOP4 contains the following information:

```
Norms: known(surface1,incline-angle)
       known(body1,acceleration-y-axis)
       desired(body1,acceleration-x-axis)
       feature(inclined,surface1)   relation(on(body1,surface1))
       object(surface,surface1)   force(body1,gravity-x-axis)
       force(body1,normal-y-axis)
Inferences:
       If a body is on an object
       Then there is a normal force acting on the body.
Solution Method:  Method-A (Force)
                  (body1,acceleration-x-axis)
                  Eqn-F3    ↓
                  (body1,sum-of-forces-x-axis)
                  Eqn-F1    ↓
                  (body1,gravity-x-axis)
                  Eqn-F5
```

Indices in Figure 5.3 are now shown beside a single line leading to stored problems and P-MOPs. For instance, eight *different* indices lead to MOP4, one of which is feature = inclined surface1.

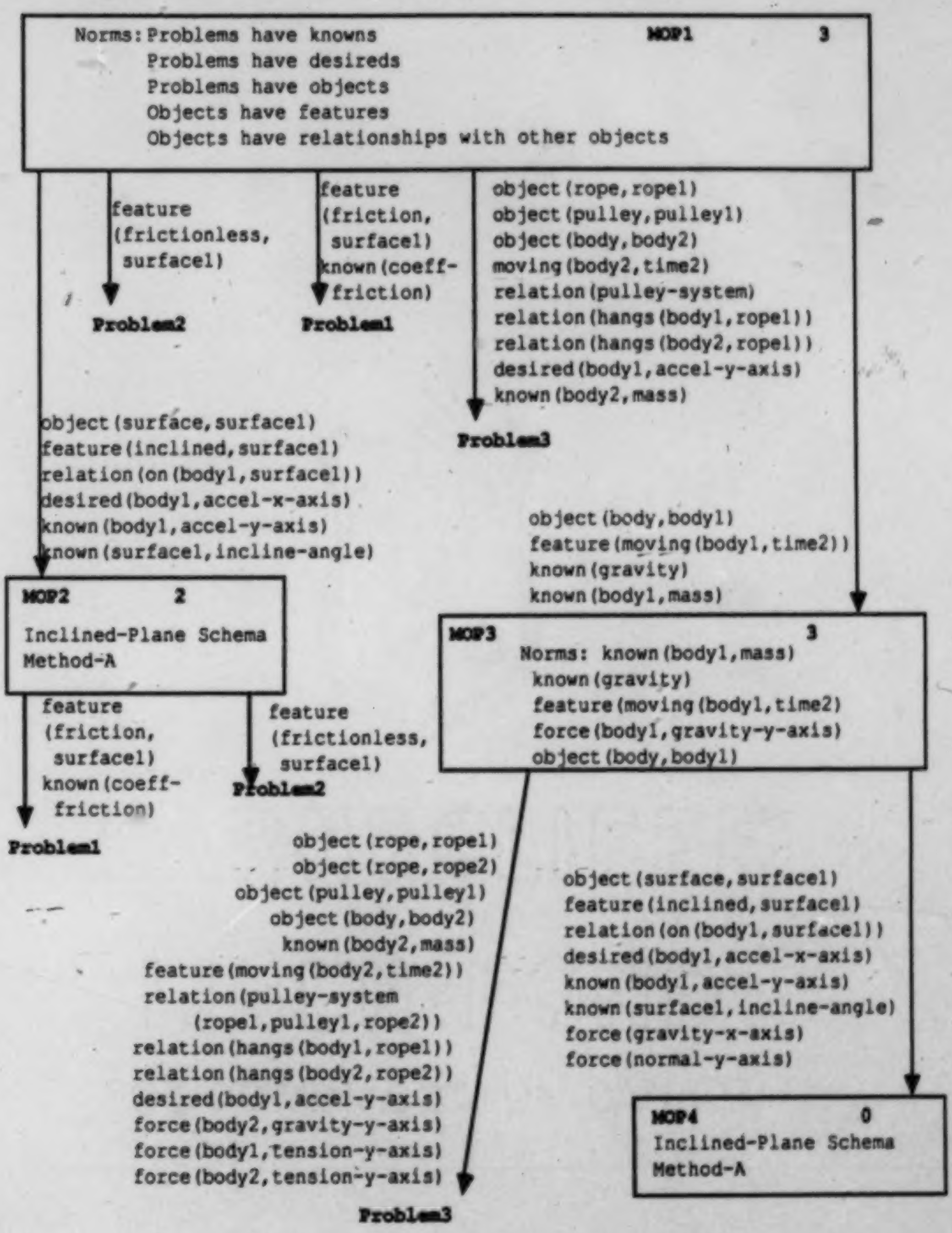

Norms: Problems have knowns
Problems have desireds
Problems have objects
Objects have features
Objects have relationships with other objects
MOP1
3
feature (frictionless, surface1)
Problem2
feature (friction, surface1)
known (coeff-friction)
Problem1
object (rope, rope1)
object (pulley, pulley1)
object (body, body2)
moving (body2, time2)
relation (pulley-system)
relation (hangs (body1, rope1))
relation (hangs (body2, rope1))
desired (body1, accel-y-axis)
known (body2, mass)
Problem3
object (surface, surface1)
feature (inclined, surface1)
relation (on (body1, surface1))
desired (body1, accel-x-axis)
known (body1, accel-y-axis)
known (surface1, incline-angle)
object (body, body1)
feature (moving (body1, time2))
known (gravity)
known (body1, mass)
MOP2
2
Inclined-Plane Schema
Method-A
MOP3
3
Norms: known (body1, mass)
known (gravity)
feature (moving (body1, time2)
force (body1, gravity-y-axis)
object (body, body1)
feature (friction, surface1)
known (coeff-friction)
Problem1
feature (frictionless, surface1)
Problem2
object (rope, rope1)
object (rope, rope2)
object (pulley, pulley1)
object (body, body2)
known (body2, mass)
feature (moving (body2, time2))
relation (pulley-system (rope1, pulley1, rope2))
relation (hangs (body1, rope1))
relation (hangs (body2, rope2))
desired (body1, accel-y-axis)
force (body2, gravity-y-axis)
force (body1, tension-y-axis)
force (body2, tension-y-axis)
Problem3
object (surface, surface1)
feature (inclined, surface1)
relation (on (body1, surface1))
desired (body1, accel-x-axis)
known (body1, accel-y-axis)
known (surface1, incline-angle)
force (gravity-x-axis)
force (normal-y-axis)
MOP4
0
Inclined-Plane Schema
Method-A

Figure 5.3 P-MOP network after Problem3 is indexed

Problem 4:

A 4.0 kg block starts up a 30 degree incline with 128 joules of kinetic energy. How far will it slide up the plane if the coefficient of friction is 0.30?

Solution:

EUREKA shows a preference in its traversal of the P-MOP network for the first time. MOP3 and its sub-MOPs are visited before MOP2 because MOP3 organizes the most problem-solving experiences. MOP3 contains no inferences, but MOP4 is reached via several indices associated with inclined plane problems. MOP4 provides a relevant inference:

```
If a body is on an object
Then there is a normal force acting on the body.
```

When problem1 is finally visited, the following inference dealing with friction is used:

```
If a body is on a surface and the surface has friction involved
Then there is a friction force acting on the body.
```

This is the first time that EUREKA finds inferences in the P-MOP network that both match patterns in the current problem and are actually helpful to the solution. These two inferences are helpful because they indicate all the relevant forces that must be considered in order to calculate work due to friction. Because these forces are found in the P-MOP network, EUREKA does not have to search for them when it uses means-ends analysis to solve the problem.

P-MOP Reorganization:

Nine new MOPs are created. The P-MOP network is growing quickly, therefore the specific problems that are indexed at various levels of the network will no longer be shown.

The current problem, Problem4, is an energy problem involving an inclined plane. When Problem4's enhanced representation is indexed into the P-MOP network, MOP2 is reached via five of its indices (the index desired = (body1,acceleration-x-axis) is not traversed because the current problem requires the distance travelled). Two of MOP2's norms do not appear as features in Problem4's enhanced representation (i.e., feature (moving(body1,time2)) and desired(body1,acceleration-x-axis)). Also the solution method, Method-A, on MOP2 has nothing in common with the solution method of the current problem. As a result MOP2 changes into MOP10. Thus, MOP10 evolves (from MOP2) as an inclined plane schema organizing two sub-MOPs, MOP11 and MOP12. MOP10 contains the following information:

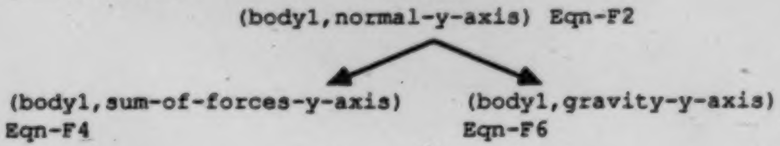
```
Norms: known(body1,mass)  known(gravity)
       known(surface1,incline-angle)
       known(body1,acceleration-y-axis)
       feature(inclined,surface1)  relation(on(body1,surface1))
       object(body,body1) object(surface,surface1)
       force(body1,gravity-x-axis)  force(body1,gravity-y-axis)
       force(body1,normal-y-axis)
Inferences:
       If a body is on an object
       Then there is a normal force acting on the body.
```

MOP11 holds Method-A and the two norms that did not appear in Problem4:

```
Norms: desired(body1,acceleration-x-axis)
       feature(moving(body1,time2))
Solution Method:  Method-A (Force)
                  (body1,acceleration-x-axis)
                  Eqn-F3      ↓
                  (body1,sum-of-forces-x-axis)
                  Eqn-F1      ↓
                  (body1,gravity-x-axis)
                  Eqn-F5
```

MOP12 is created as a result of finding commonalities between an inclined plane problem solved with force methods, Problem1, and the current problem. Both of these problems involved friction and had a common sub-portion in their solution methods (Method-B). MOP12 will be referred to as a friction schema and contains the following information:

```
Norms: known(coefficient-friction)  feature(friction,surface1)
       force(body1,friction-x-axis)
Inferences:
       If a body is on an object
       Then there is a normal force acting on the body.
       If a body is on a surface and the surface has friction
       Then there is a friction force acting on the body.
Solution Method:  Method-B (Force)
```

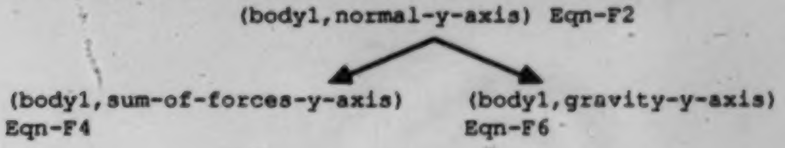


Therefore, as a result of integrating the current problem into the P-MOP network, the knowledge that resides in MOP10, MOP11 and MOP12 reflects a more sophisticated organization of inferences, methods and commonalites of what was previously organized on one MOP, MOP2.

A similar, but more complex, restructuring of knowledge has happened with MOPs 5 through 9. The knowledge previously organized on three of the four occurrences of MOP3 and its substructure is redistributed in MOPs 5 through 9 as a result of integrating the current problem into the P-MOP network.

MOP13, like MOP12, reflects the commonalities between Problem1, an inclined plane problem involving friction solved with force methods, and the current problem. MOP13 is indexed off the top MOP with friction indices and contains the common abstracted sub-method (Method-B). Notice that MOP13, shown below, is similar to MOP12. Because MOP13 sits higher in the network, it also includes norms dealing with inclined planes.

```
Norms: known(body1,mass)  known(gravity)
       known(surface1,incline-angle)  known(coefficient-friction)
       known(body1,acceleration-y-axis)
       feature(inclined,surface1)  feature(friction,surface1)
       relation(on(body1,surface1)) object(body,body1)
       object(surface,surface1)  force(body1,gravity-x-axis)
       force(body1,gravity-y-axis) force(body1,normal-y-axis)
       force(body1,friction-x-axis)
Inferences:
      If a body is on an object
      Then there is a normal force acting on the body.
      If a body is on a surface and the surface has friction
      Then there is a friction force acting on the body.
Solution Method:  Method-B (Force)
```



All of the above P-MOPs also contain differences that are indices to the problems indexed under the P-MOP.

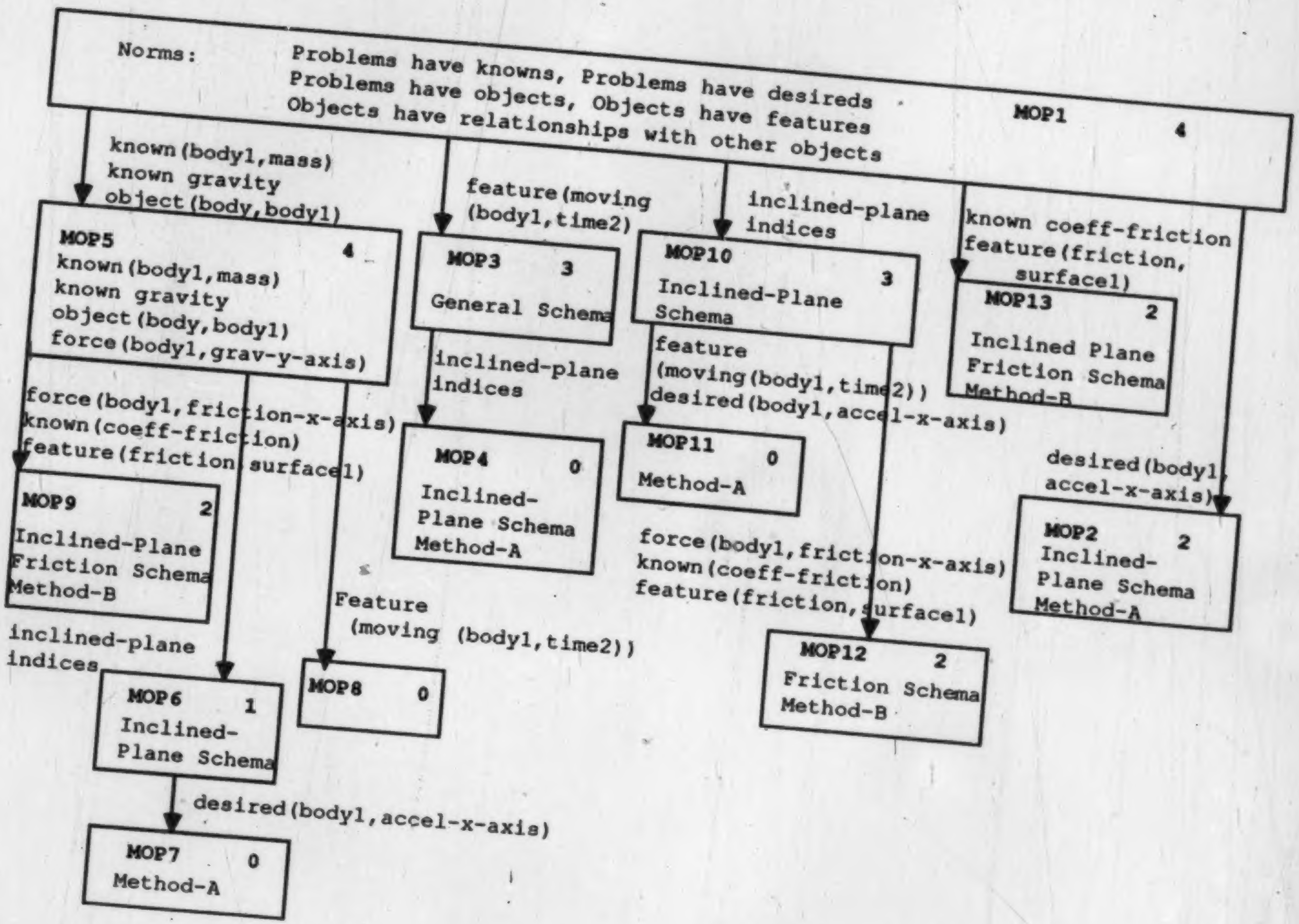


Figure 5.4 P-MOP network after Problem4 is indexed

Problem 5:

A 100 kg. Man (body1) lowers himself to the ground from a height of 10 meters by means of a rope passed over a frictionless, massless pulley and attached to a 70 kg. sandbag (body2). What is the kinetic energy of the man as he hits the ground?

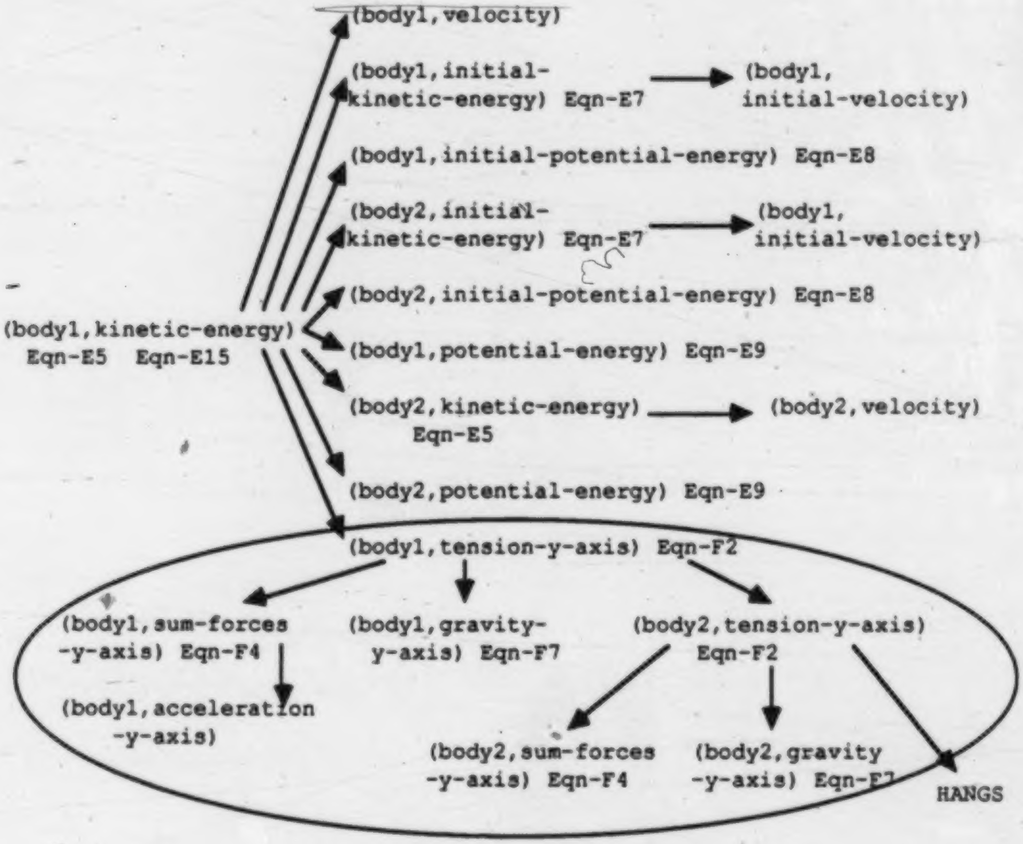
Solution:

The only P-MOPs traversed by EUREKA are MOP5 and MOP8, followed by MOP3. These MOPs do not contain inferences. When problem3, EUREKA's only pulley problem experience, is found, its inferences are used. These inferences are:

```
If a body is hanging from a rope
Then there is a force due to tension acting on the body.

If a body hangs from a rope
      and another body hangs from another rope
      and these two ropes and a pulley form a pulley-system
Then the tension force acting on the two bodies are equal
      and the acceleration of the two bodies are equal.
```

EUREKA finds these inferences during its directed traversal of the P-MOP network because of surface similarities between this problem and the one pulley problem indexed in the network. However, these inferences are irrelevant to the current problem and they lead EUREKA off on an irrelevant tangent during its solution. Because the means-ends strategy focuses on the desired quantity (in this case, kinetic-energy), EUREKA started solving this problem correctly by retrieving energy equations. However, when it ran out of energy equations and still did not have the answer, EUREKA looked for another lead to follow. The next focus-of-attention in these cases is a recently inferred concept. For this problem, the (irrelevant inference) about the presence of tension caused EUREKA to focus on tension-along-the-y-axis. The circled steps of the solution structure below correspond to this irrelevant tangent. EUREKA eventually solves this problem because it runs out of (irrelevant) force equations and newly-inferred concepts. Relations among objects in the problem is EUREKA's next possible lead and therefore "hangs" becomes the next focus-of-attention. This relation leads to the following inference which is necessary for the solution of the problem:

```
If a body is hanging from a rope and the body is at rest at time1
Then the Initial Kinetic Energy of the body is known.
```


(body1, velocity)
(body1, initial-kinetic-energy) Eqn-E7
(body1, initial-velocity)
(body1, initial-potential-energy) Eqn-E8
(body2, initial-kinetic-energy) Eqn-E7
(body1, initial-velocity)
(body2, initial-potential-energy) Eqn-E8
(body1, kinetic-energy) Eqn-E5 Eqn-E15
(body1, potential-energy) Eqn-E9
(body2, kinetic-energy) Eqn-E5
(body2, velocity)
(body2, potential-energy) Eqn-E9
(body1, tension-y-axis) Eqn-F2
(body1, sum-forces -y-axis) Eqn-F4
(body1, gravity-y-axis) Eqn-F7
(body2, tension-y-axis) Eqn-F2
(body1, acceleration -y-axis)
(body2, sum-forces -y-axis) Eqn-F4
(body2, gravity -y-axis) Eqn-F7
HANGS

Solution method for Problem5

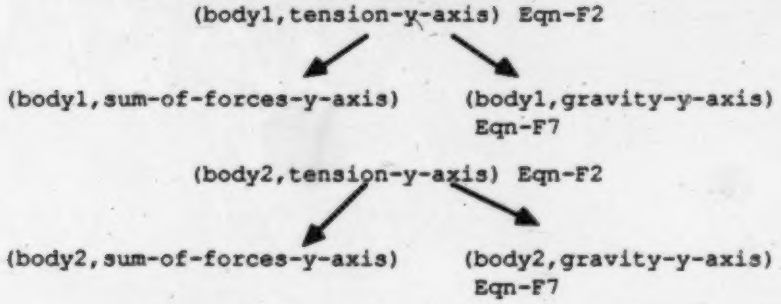
Note that (body1, velocity) at the top of the solution method tree has no equations associated with it. What this means in this diagram and all subsequent diagrams of solution methods is that there were no appropriate equations (i.e., whose constraints were met and had the correct number of unknowns) to solve for this quantity.

P-MOP Reorganization:

Figure 5.5 shows that, after problem5's enhanced representation is indexed, five MOPs disappear and three new MOPs are created. MOP5 is stable and indexes a majority of its parent MOP's problems. Therefore, it collapses into MOP1 (the root mop). MOP5 and its sub-MOPs disappear. Note that the information contained in MOP5 and its sub-MOPs appear elsewhere, at a higher level, in the resulting network. MOP1 now contains the information that physics problems involve a body, the mass of the body is known, gravity is a known quantity, and a force due to gravity acts on the body along the y-axis. Now that force is a norm on MOP1, future problems will be indexed under MOP1 by the forces that occur in the problem. The force index can be regarded as a "deep" feature because the forces acting on bodies are not explicit in the problem statement, but rather are determined by using inferences. Thus, a deep feature has evolved as an index under the top-most P-MOP.

The P-MOP network of Figure 5.5 now contains a pulley schema, MOP15, with a new abstracted solution method, Method-C. MOP15 , shown below, evolved because of commonalities the current problem had with the only other pulley problem indexed in the network, Problem3. These two problems shared a number of features, inferences, and a portion of their solution methods. Note that a number of these commonalities, including the abstracted solution sub-method, are as a result of Problem5's irrelevant tangent concerning tensions and forces. When EUREKA solves more energy pulley problems it will discover that Method-C is not a common solution sub-method to all pulley problems. Therefore, this method will eventually be removed from MOP15.

```
Norms: known(body2,mass)   relation(hangs(body1,rope1))
       relation(hangs(body2,rope2))
       relation(pulley-system(rope1,pulley1,rope2))
       object(rope,rope1)  object(rope,rope2)
       object(pulley,pulley1)  object(body,body2)
       feature(moving(body1,time2))  feature(moving(body2,time2))
       force(body2,gravity-y-axis)  force(body1,tension-y-axis)
       force(body2,tension-y-axis)
       equal-to((body1,tension-y-axis)(body2,tension-y-axis))
       equal-to((body1,acceleration-y-axis)
                (body2,acceleration-y-axis))
Inferences:
      If a body is hanging from a rope
      Then there is a force due to tension acting on the body.
      If a body hangs from a rope
            and another body hangs from another rope
            and these two ropes and a pulley form a pulley-system
      Then the tension force acting on the two bodies are equal
            and the acceleration of the two bodies are equal.
      Solution Method:  Method-C (Force)
```


(body1,tension-y-axis) Eqn-F2
(body1,sum-of-forces-y-axis)
(body1,gravity-y-axis)
Eqn-F7
(body2,tension-y-axis) Eqn-F2
(body2,sum-of-forces-y-axis)
(body2,gravity-y-axis)
Eqn-F7

The Energy Schema, MOP16, contains features that were common between an inclined plane problem solved with energy methods, Problem4, and this pulley problem, also solved with energy methods. MOP16 contains the following information:

```
Norms: known(body1,initial-kinetic-energy)
       known(body1,initial-height)
```

There are no inferences or solution method common to both the energy incline-plane problem and the energy pulley problem.

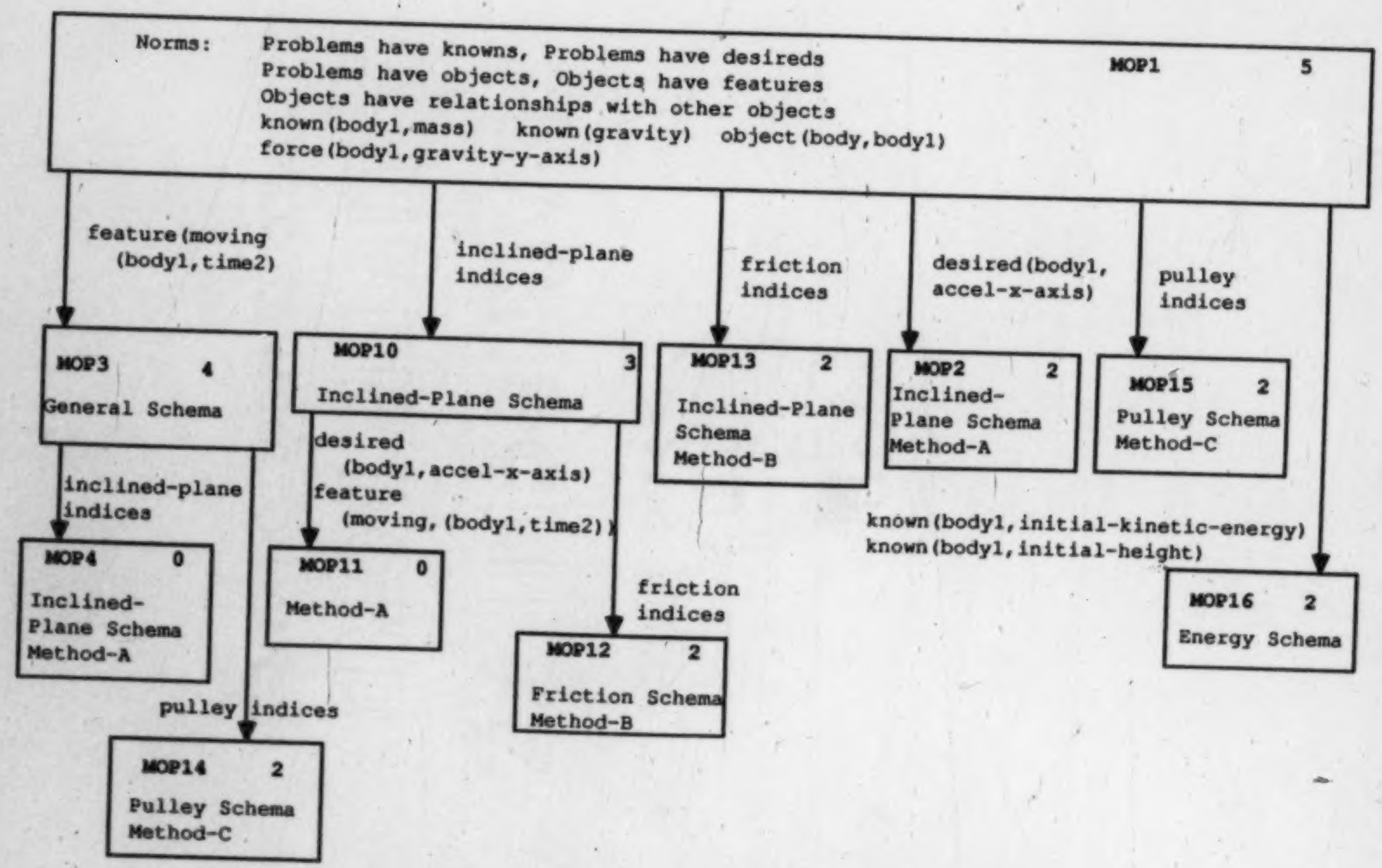

Norms:
Problems have knowns, Problems have desireds
Problems have objects, Objects have features
Objects have relationships with other objects
known(body1,mass) known(gravity) object(body,body1)
force(body1,gravity-y-axis)
MOP1 5
feature(moving (body1,time2)
inclined-plane indices
friction indices
desired(body1, accel-x-axis)
pulley indices
MOP3 4
General Schema
MOP10 3
Inclined-Plane Schema
MOP13 2
Inclined-Plane Schema Method-B
MOP2 2
Inclined-Plane Schema Method-A
MOP15 2
Pulley Schema Method-C
inclined-plane indices
desired (body1,accel-x-axis)
feature (moving, (body1,time2))
known(body1,initial-kinetic-energy)
known(body1,initial-height)
MOP4 0
Inclined-Plane Schema Method-A
MOP11 0
Method-A
friction indices
MOP16 2
Energy Schema
MOP12 2
Friction Schema Method-B
pulley indices
MOP14 2
Pulley Schema Method-C

Figure 5.5 P-MOP network after Problem5 is indexed

Problem 6:

A x kg. sandbag (body1) is on the ground and is attached to a rope passed over a frictionless, massless pulley holding a y kg. man (body2). The man lowers himself to the ground from a height of z meters. What is the kinetic energy of the man as he hits the ground?

Solution:

EUREKA first traverses MOP3 via the index feature = (moving (body1, time2)). This path is preferred because this pattern has occurred most frequently for EUREKA so far. MOP3 did not contain any inferences and futher traversal from MOP3 along pulley indices brings EUREKA to MOP14. EUREKA finds these inferences on MOP14:

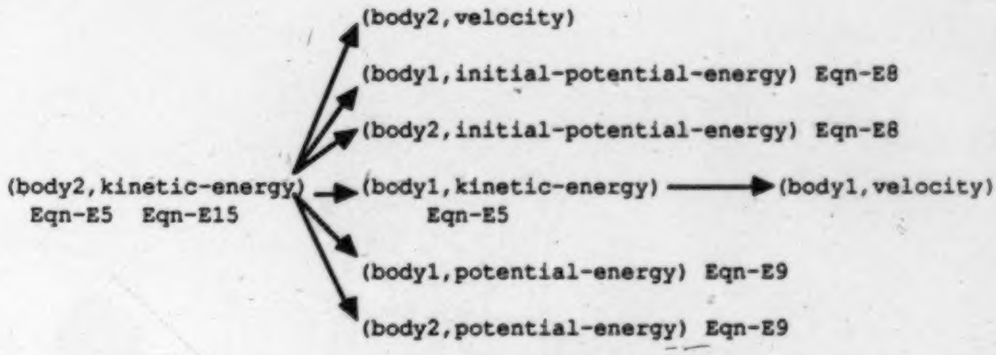
```
If a body is hanging from a rope
Then there is a force due to tension acting on the body.

If a body hangs from a rope
      and another body hangs from another rope
      and these two ropes and a pulley form a pulley-system
Then the tension force acting on the two bodies are equal
      and the acceleration of the two bodies are equal.
```

Further traversal from MOP14 brings EUREKA to problem5 which contains the following inference dealing with initial kinetic energy:

```
If a body is hanging from a rope and the body is at rest at time1
Then the Initial Kinetic Energy of the body is known.
```

The same inferences about ropes and tensions were found on MOP15 at a later stage of the traversal. EUREKA also reaches MOP16, but finds no inferences there. MOP16 was created as a result of an energy inclined plane problem and an energy pulley problem that did not have any common inferences.

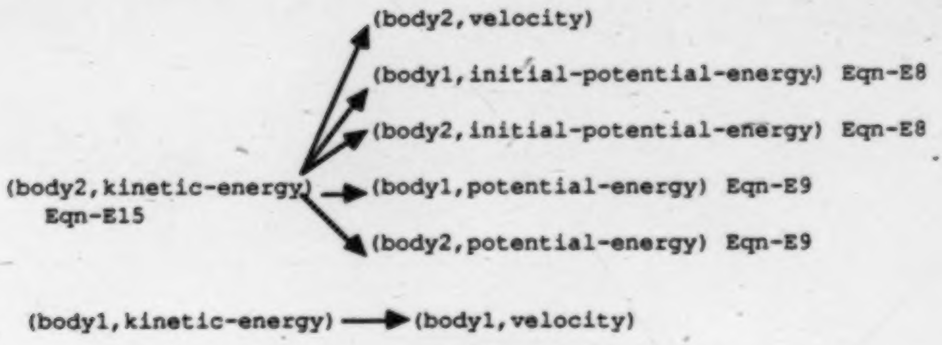
The inference regarding initial kinetic energy, found in the early stages of the P-MOP traversals, was the critical inference for the current problem. Because this inference was found in the P-MOP network rather than searched for during means-ends analysis, EUREKA's solution to problem6, unlike the problem5 solution, did not go off on a force side-track. Retrieving this inference from its past problem-solving experiences enabled EUREKA to infer the initial kinetic energies of the bodies in the problem. Unlike its solution to Problem5, EUREKA was able to solve the problem before running out of equations to consider and therefore considering (irrelevant) force concepts. The solution method of problem6 is:


(body2,velocity)
(body1,initial-potential-energy) Eqn-E8
(body2,initial-potential-energy) Eqn-E8
(body2,kinetic-energy)
Eqn-E5 Eqn-E15
(body1,kinetic-energy)
Eqn-E5
(body1,velocity)
(body1,potential-energy) Eqn-E9
(body2,potential-energy) Eqn-E9

Solution method for Problem6

P-MOP Reorganization:

The first energy solution method, Method-D, shows up in the P-MOP network because indexing the enhanced problem6 representation has commonalities with problem5. Five new MOPs are created (i.e., MOP17, MOP18, MOP19, MOP20 and MOP21) because of the current problems commonalities with Problem5 and all of these new MOPs contain the new abstracted energy solution method:


(body2,velocity)
(body1,initial-potential-energy) Eqn-E8
(body2,initial-potential-energy) Eqn-E8
(body2,kinetic-energy)
Eqn-E15
(body1,potential-energy) Eqn-E9
(body2,potential-energy) Eqn-E9
(body1,kinetic-energy)
(body1,velocity)

Method-D (Energy)

When problem6 traversed the two MOPs (i.e., MOP14 and MOP15) that contained Method-C (the percieved common solution method between a force pulley problem, Problem3, and the energy pulley problem, Problem5, that went off on a force side-track during its solution), this method was removed from these MOPs because it was not part of problem6's solution. MOP14 and MOP15 are now pulley schemas that index sub-MOPs containing solution methods. Note that this sub-structure indicates that there is no solution method common to all pulley problems. Additional constraints must be met in order to reach the solution method on the sub-MOP's (i.e., MOP17 or MOP19) or the stored problems.

MOP1 contains a predictive feature, feature(moving(body1,time2)), which indexes MOP3. MOP3 is stable and contains norms that co-occur. Therefore, its indexing feature, feature(moving(body1,time2)), is marked as predictive and generally regarded now as a "good" feature on which to discriminate future problems.

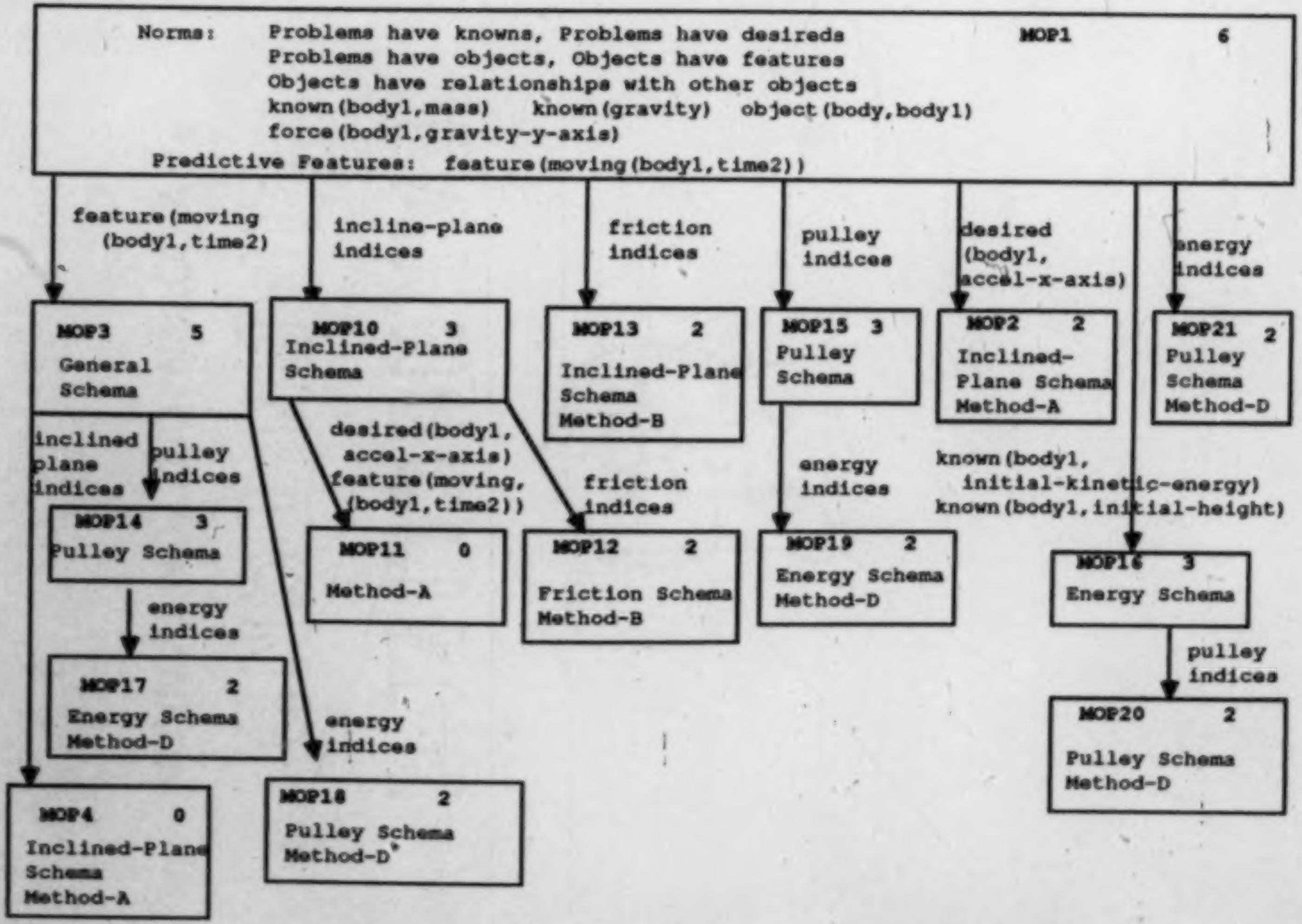

Norms:
Problems have knowns, Problems have desireds
Problems have objects, Objects have features
Objects have relationships with other objects
known(body1,mass) known(gravity) object(body,body1)
force(body1,gravity-y-axis)
Predictive Features: feature(moving(body1,time2))
MOP1 6
feature(moving (body1,time2)
incline-plane indices
friction indices
pulley indices
desired (body1, accel-x-axis)
energy indices
MOP3 5 General Schema
MOP10 3 Inclined-Plane Schema
MOP13 2 Inclined-Plane Schema Method-B
MOP15 3 Pulley Schema
MOP2 2 Inclined-Plane Schema Method-A
MOP21 2 Pulley Schema Method-D
inclined plane indices
pulley indices
desired(body1, accel-x-axis) feature(moving, (body1,time2))
friction indices
energy indices
known(body1, initial-kinetic-energy) known(body1,initial-height)
MOP14 3 Pulley Schema
MOP11 0 Method-A
MOP12 2 Friction Schema Method-B
MOP19 2 Energy Schema Method-D
MOP16 3 Energy Schema
energy indices
MOP17 2 Energy Schema Method-D
energy indices
pulley indices
MOP20 2 Pulley Schema Method-D
MOP4 0 Inclined-Plane Schema Method-A
MOP18 2 Pulley Schema Method-D*

Figure 5.6 P-MOP network after Problem6 is indexed

Problem 7:

A body of mass 5 kgs starts with 50 joules of energy from the top of an inclined plane of length 10 meters. The plane is inclined 30 degrees from the horizontal and the coefficient of friction is 0.30. What is the velocity of the body when it reaches the bottom of the plane?

Solution:

The features of this problem match several indices in the P-MOP network of Figure 5.6. During its traversal, EUREKA finds the following force inferences on the MOPs:

```
If a body is on a surface and the surface has friction involved
Then there is a friction force acting on the body.

If a body is on an object
Then there is a normal force acting on the body.
```

None of the MOPs traversed contain energy inferences relevant to the solution of this problem. However, EUREKA has seen this problem type before, in problem4, and finds the following inference in the stored representation of problem4:

```
If a body is on an object and is at rest at time2
Then the Kinetic Energy of the body is known.
```

Because EUREKA has inferred the information that the kinetic energy of the body is known, it does not have to search for equations to determine this quantity when it uses means-ends analysis to solve the problem. EUREKA has found all the inferences necessary for the solution of this problem in the P-MOP network.

P-MOP Reorganization:

Quite a few new MOPs are created. The P-MOP network of Figure 5.7 now contains pulley and inclined plane MOPs with energy sub-MOPs. For example, MOP13 is reached by inclined plane surface features, (i.e., feature = (friction,surface1)), which leads to a sub-organization of problem types organized according to their underlying principles (i.e., MOP32 is indexed by energy features such as known = (body1, initial-kinetic-energy)). Similarily MOP15, reached via pulley surface features, has a sub-organization of energy problem types containing a solution sub-method (i.e., Method-D) that was common to the pulley problems it has solved with energy methods.

The P-MOP network also contains a few instances of P-MOP's representing a physics principle with a sub-organization of surface feature sub-MOPs. For example, MOP16 and MOP39 are energy P-MOPs (reached via energy-like indices) with pulley sub-MOPs (i.e., reached via pulley surface feature indices). MOP16 contains the following information:

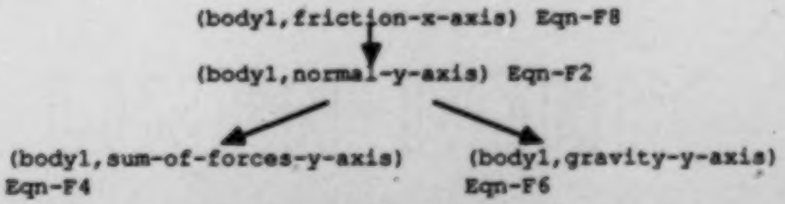
```
Norms: known(body1,initial-height)
       known(body1,initial-kinetic-energy)
```

MOP39 contains:

```
Norms: known(body1,height)
       known(body1,initial-kinetic-energy)
       feature(moving(body1,time2))
```

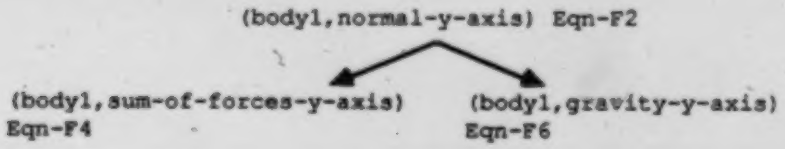
The P-MOP network is beginning to move towards a more expert-like organization in which physics problems are organized according to their underlying physics principles.

The following new solution dealing with friction appears in the P-MOP network on MOP26, MOP30 and MOP33:


(body1,friction-x-axis) Eqn-F8
(body1,normal-y-axis) Eqn-F2
(body1,sum-of-forces-y-axis) Eqn-F4
(body1,gravity-y-axis) Eqn-F6

Method-E (Friction)

Method-E is similar to the previously abstracted Method-B as shown below. The difference is that Method-E contains an additional step which makes it a more complete solution for friction. Method-B simply solves for normal-force without detailing the goal that normal-force was solved for. EUREKA's solution abstraction process abstracts common sub-methods and also refines and enlarges these sub-methods.



Method-B (Force)

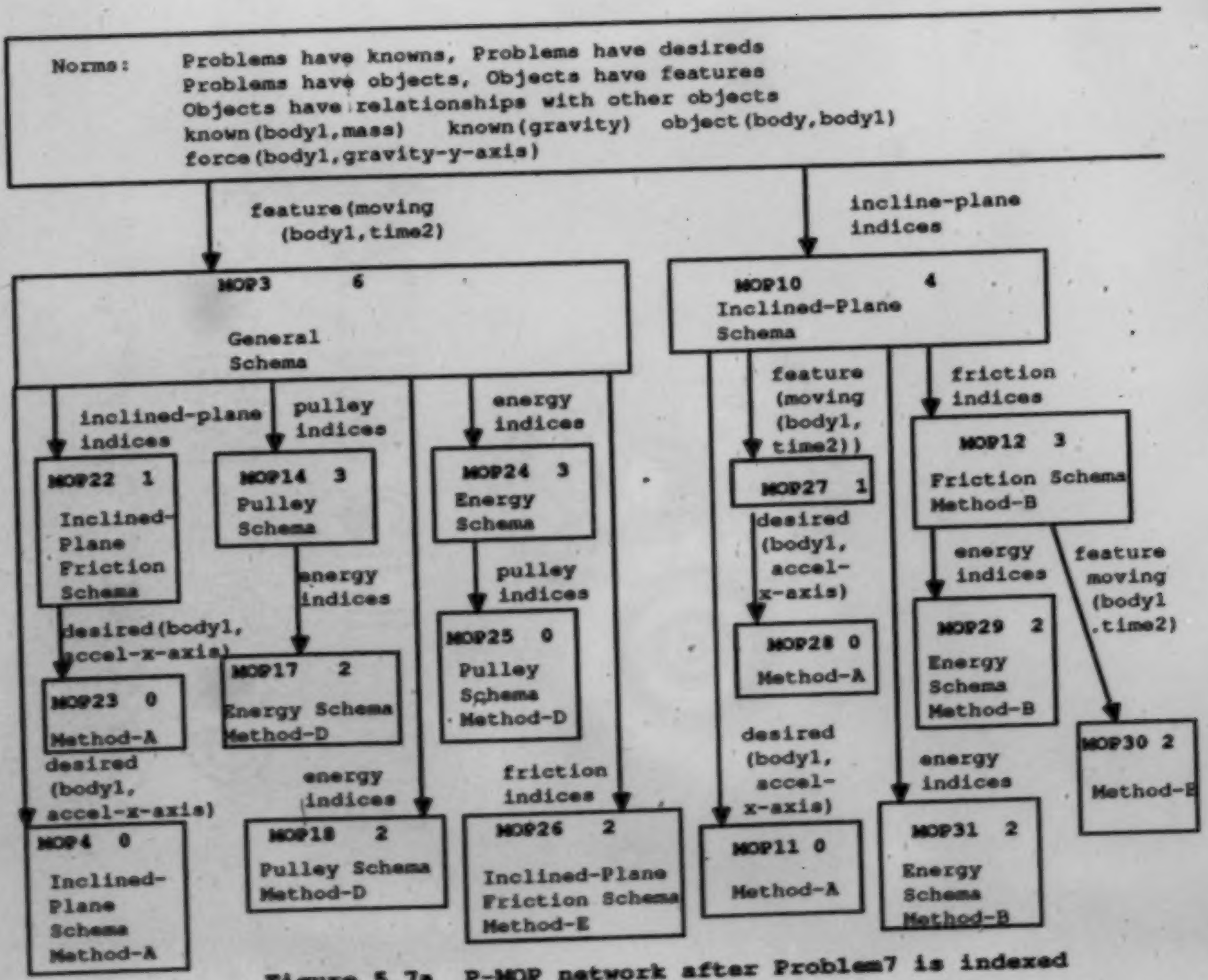


Figure 5.7a P-MOP network after Problem7 is indexed

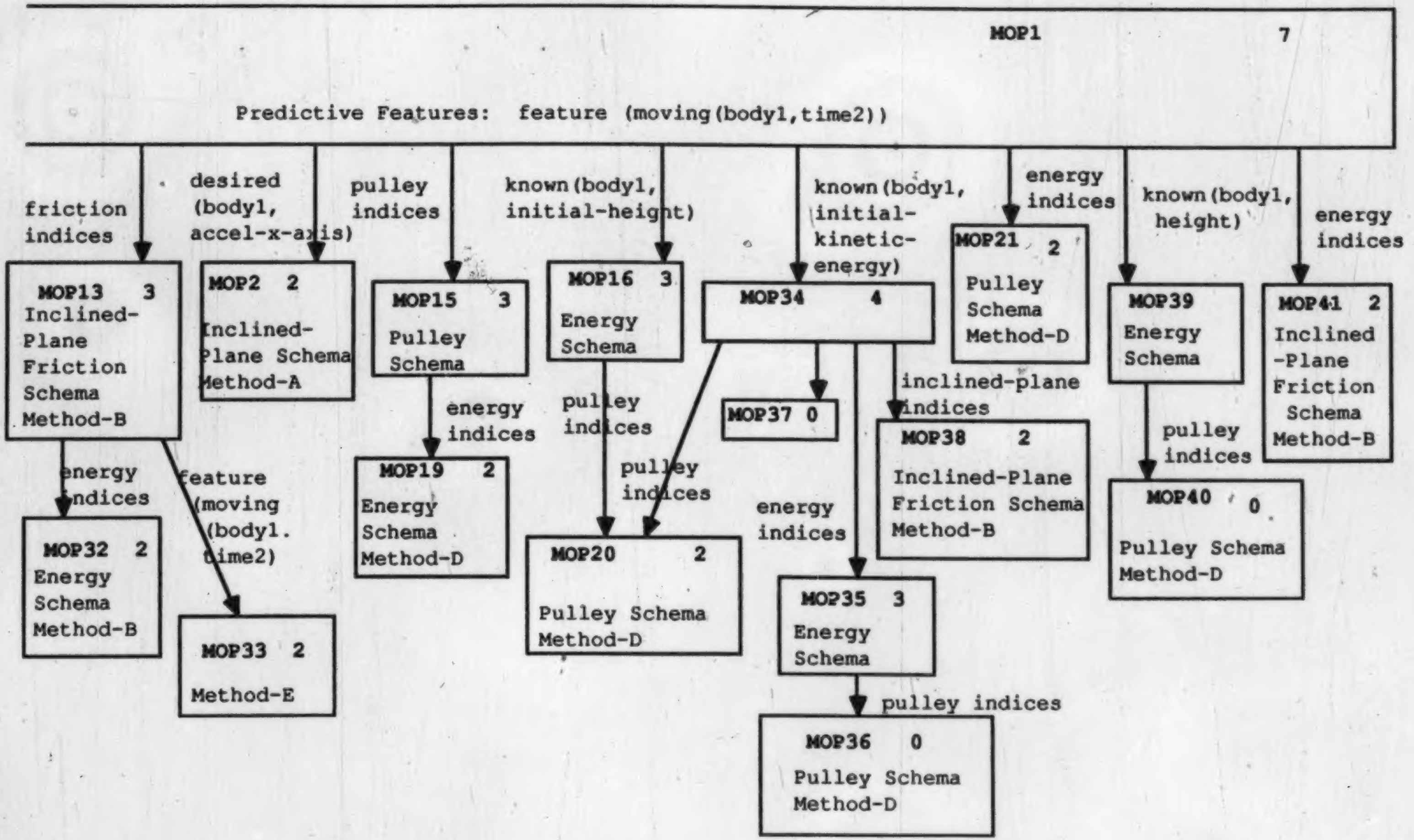

MOP1 7
Predictive Features: feature (moving(body1,time2))
friction indices
desired (body1, accel-x-axis)
pulley indices
known(body1, initial-height)
known(body1, initial-kinetic-energy)
energy indices
known(body1, height)
energy indices
MOP13 3 Inclined-Plane Friction Schema Method-B
MOP2 2 Inclined-Plane Schema Method-A
MOP15 3 Pulley Schema
MOP16 3 Energy Schema
MOP34 4
MOP21 2 Pulley Schema Method-D
MOP39 Energy Schema
MOP41 2 Inclined-Plane Friction Schema Method-B
energy indices
feature (moving (body1. time2)
energy indices
pulley indices
pulley indices
inclined-plane indices
energy indices
pulley indices
MOP32 2 Energy Schema Method-B
MOP33 2 Method-E
MOP19 2 Energy Schema Method-D
MOP20 2 Pulley Schema Method-D
MOP37 0
MOP38 2 Inclined-Plane Friction Schema Method-B
MOP35 3 Energy Schema
MOP40 0 Pulley Schema Method-D
pulley indices
MOP36 0 Pulley Schema Method-D

Figure 5.7b P-MOP network after Problem7 is indexed

Problem 8:

Consider two unequal masses connected by a rope which passes over a frictionless, massless pulley. The mass of body1 is x slug and the mass of body2 is y slugs. The bodies have an acceleration of z ft/sec. What is the tension of the rope?

Solution:

Because MOP3 indexes the most problem solving experiences, it is visited first and the pulley features that index MOP14 from MOP3 are matched by this problem. EUREKA is able to find the following inferences on MOP14:

```
If a body is hanging from a rope
Then there is a force due to tension acting on the body.

If a body hangs from a rope
      and another body hangs from another rope
      and these two ropes and a pulley form a pulley-system
Then the tension force acting on the two bodies are equal
      and the acceleration of the two bodies are equal.
```

Again, all relevant inferences are found from past problem-solving knowledge, greatly simplifying the means-ends analysis solution. Moreover, these inferences are found early in the P-MOP traversal.

P-MOP Reorganization:

Four new MOPs are created. One of them, MOP45, is a new pulley schema indexed by forces that are "deep features" of pulley problems which are triggered by the problem statement. This is an example of a deep feature "moving up" the network as an important discriminating aspect of problem types.

The P-MOP network is reaching a stable state because a number of existing MOPs do not undergo any reorganization but simply have their features and solution methods confirmed by this problem.

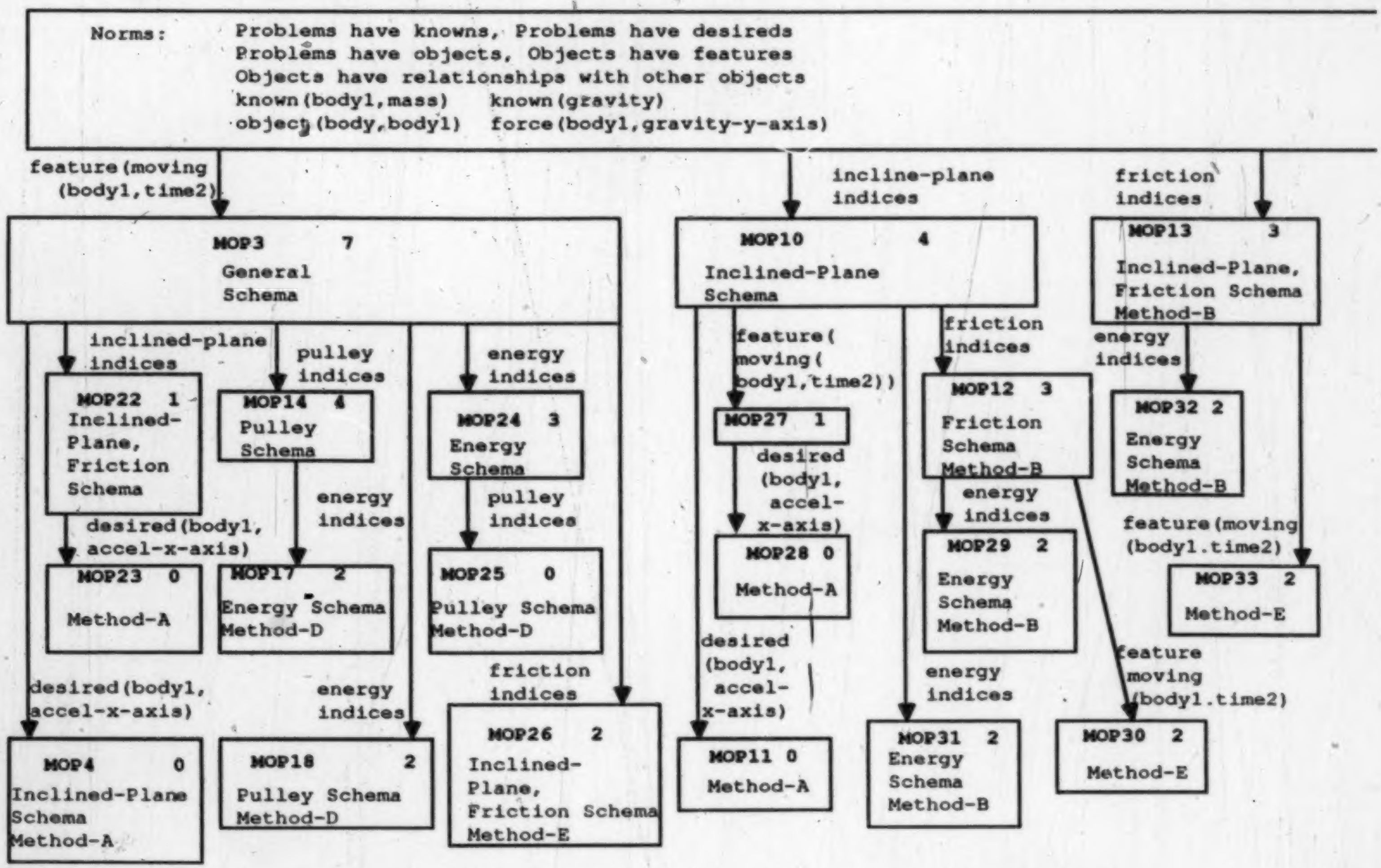

Norms:
Problems have knowns, Problems have desireds
Problems have objects, Objects have features
Objects have relationships with other objects
known(body1,mass) known(gravity)
object(body,body1) force(body1,gravity-y-axis)
feature(moving (body1,time2))
incline-plane indices
friction indices
MOP3 7 General Schema
MOP10 4 Inclined-Plane Schema
MOP13 3 Inclined-Plane, Friction Schema Method-B
inclined-plane indices
pulley indices
energy indices
MOP22 1 Inclined-Plane, Friction Schema
MOP14 4 Pulley Schema
MOP24 3 Energy Schema
desired(body1, accel-x-axis)
energy indices
pulley indices
MOP23 0 Method-A
MOP17 2 Energy Schema Method-D
MOP25 0 Pulley Schema Method-D
desired(body1, accel-x-axis)
energy indices
friction indices
MOP4 0 Inclined-Plane Schema Method-A
MOP18 2 Pulley Schema Method-D
MOP26 2 Inclined-Plane, Friction Schema Method-E
feature(moving(body1,time2))
friction indices
MOP27 1
desired (body1, accel- x-axis)
MOP28 0 Method-A
MOP12 3 Friction Schema Method-B
energy indices
MOP29 2 Energy Schema Method-B
desired (body1, accel- x-axis)
MOP11 0 Method-A
energy indices
MOP31 2 Energy Schema Method-B
energy indices
MOP32 2 Energy Schema Method-B
feature(moving (body1.time2)
MOP33 2 Method-E
feature moving (body1.time2)
MOP30 2 Method-E

Figure 5.8a P-MOP network after Problem8 is indexed

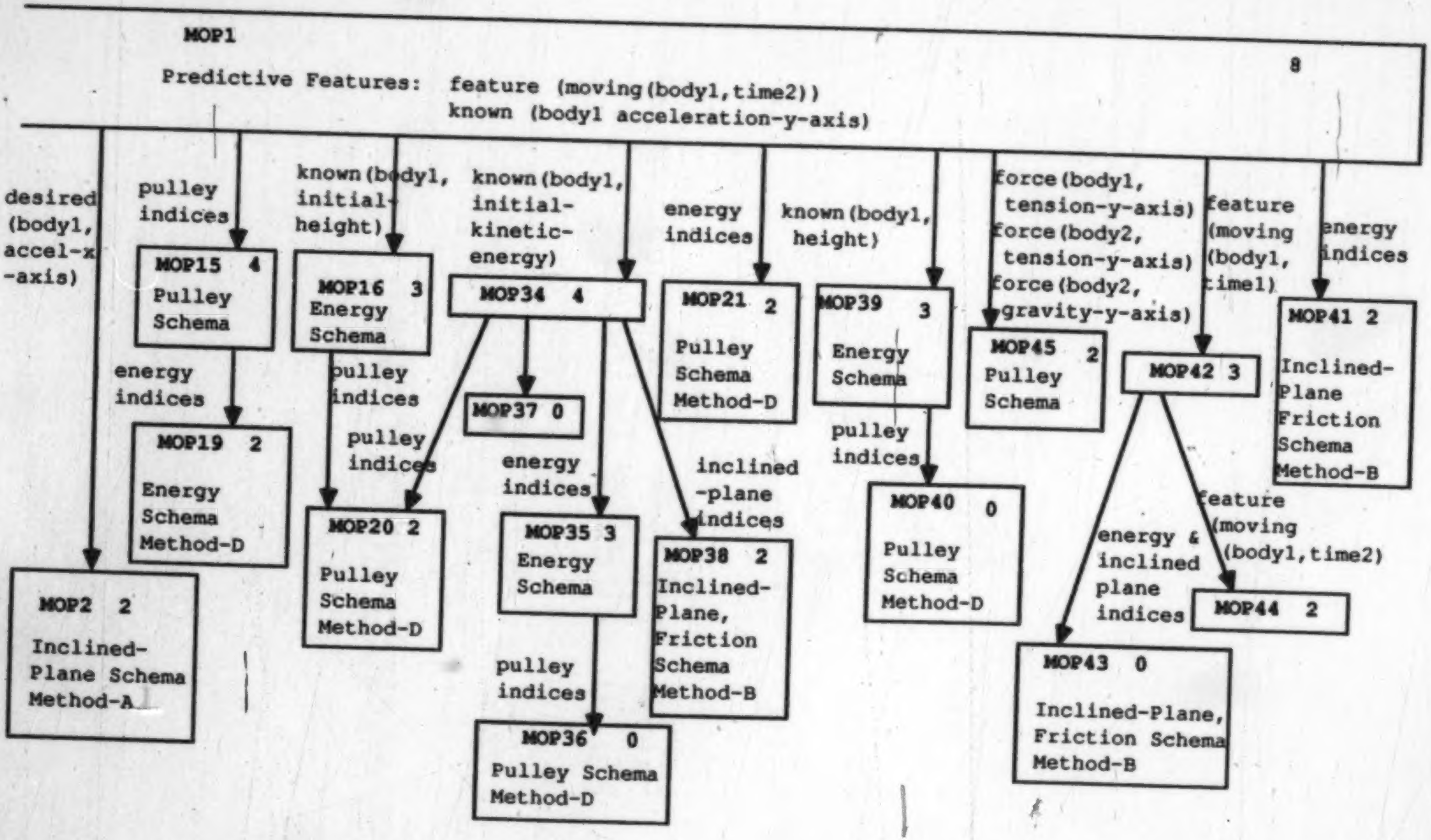

MOP1
Predictive Features: feature (moving(body1,time2))
known (body1 acceleration-y-axis)
8
desired (body1, accel-x -axis)
MOP2 2 Inclined-Plane Schema Method-A
pulley indices
MOP15 4 Pulley Schema
energy indices
MOP19 2 Energy Schema Method-D
known(body1, initial-height)
MOP16 3 Energy Schema
pulley indices
pulley indices
MOP20 2 Pulley Schema Method-D
known(body1, initial-kinetic-energy)
MOP34 4
MOP37 0
energy indices
MOP35 3 Energy Schema
pulley indices
MOP36 0 Pulley Schema Method-D
inclined -plane indices
MOP38 2 Inclined-Plane, Friction Schema Method-B
energy indices
MOP21 2 Pulley Schema Method-D
known(body1, height)
MOP39 3 Energy Schema
pulley indices
MOP40 0 Pulley Schema Method-D
force(body1, tension-y-axis)
force(body2, tension-y-axis)
force(body2, gravity-y-axis)
MOP45 2 Pulley Schema
feature (moving (body1, time1)
MOP42 3
energy & inclined plane indices
MOP43 0 Inclined-Plane, Friction Schema Method-B
feature (moving (body1,time2)
MOP44 2
energy indices
MOP41 2 Inclined-Plane Friction Schema Method-B

Figure 5.8b P-MOP network after Problem8 is indexed

Problem 9:

Consider two unequal masses connected by a rope which passes over a frictionless, massless pulley. The mass of body1 is x slug and the mass of body2 is y slugs. If the tension of the rope is z what is the acceleration of body1?

Solution:

Because MOP3 indexes the most problem solving experiences, it is visited first. As in problem 8, EUREKA is able to find all relevant inferences on MOP14 prior to using means-ends analysis to solve the problem.

P-MOP Reorganization:

MOP3 indexes a majority of its parents problem-solving experiences and therefore it collapses into MOP1. MOP3 and its sub-MOPs disappear. MOP1 now represents the additional knowledge that, in physics problems, body1 is moving at time2. Because this feature is a norm of MOP1, it is no longer eligible as a difference for traversing and indexing problems. Recall that when choosing features to traverse the P-MOP network, all norms found on a P-MOP are eliminated as indices for the current problem. This means that P-MOPs indexed by this feature (i.e., MOP27, MOP30, MOP33, etc) will no longer be traversed. Of course the feature, "body1 is moving at time2", is not generally true of all physics problems and will be removed from MOP1 when a future problem is missing this feature. When this happens, the feature again becomes eligible for use as an index and the MOPs indexed by this feature can be traversed.

Two new MOPs are created, MOP46 and MOP47. Both deal with pulleys and contain the following new sub-solution:

```
(body1,acceleration-y-axis)
Eqn-F4     ↓

(body1,sum-of-forces-y-axis)
Eqn-F2     ↓

(body1,gravity-y-axis)
Eqn-F7
```

Method-F (Force)

MOP15 organizes the most problems relative to the other MOPs indexed directly under the root MOP, MOP1. This means that any problems that contain pulley features will be drawn to MOP15 first in its traversal of the P-MOP network. However, even though MOP15 is reached by surface features, it now organizes an energy schema (MOP19) containing an energy sub-method and a force schema (MOP46) containing the above new force sub-method. The relevant solution method can be found by traversing indices that further match the current problem.

Features dealing with pulleys have become predictive and are regarded as good indices into the P-MOP network.

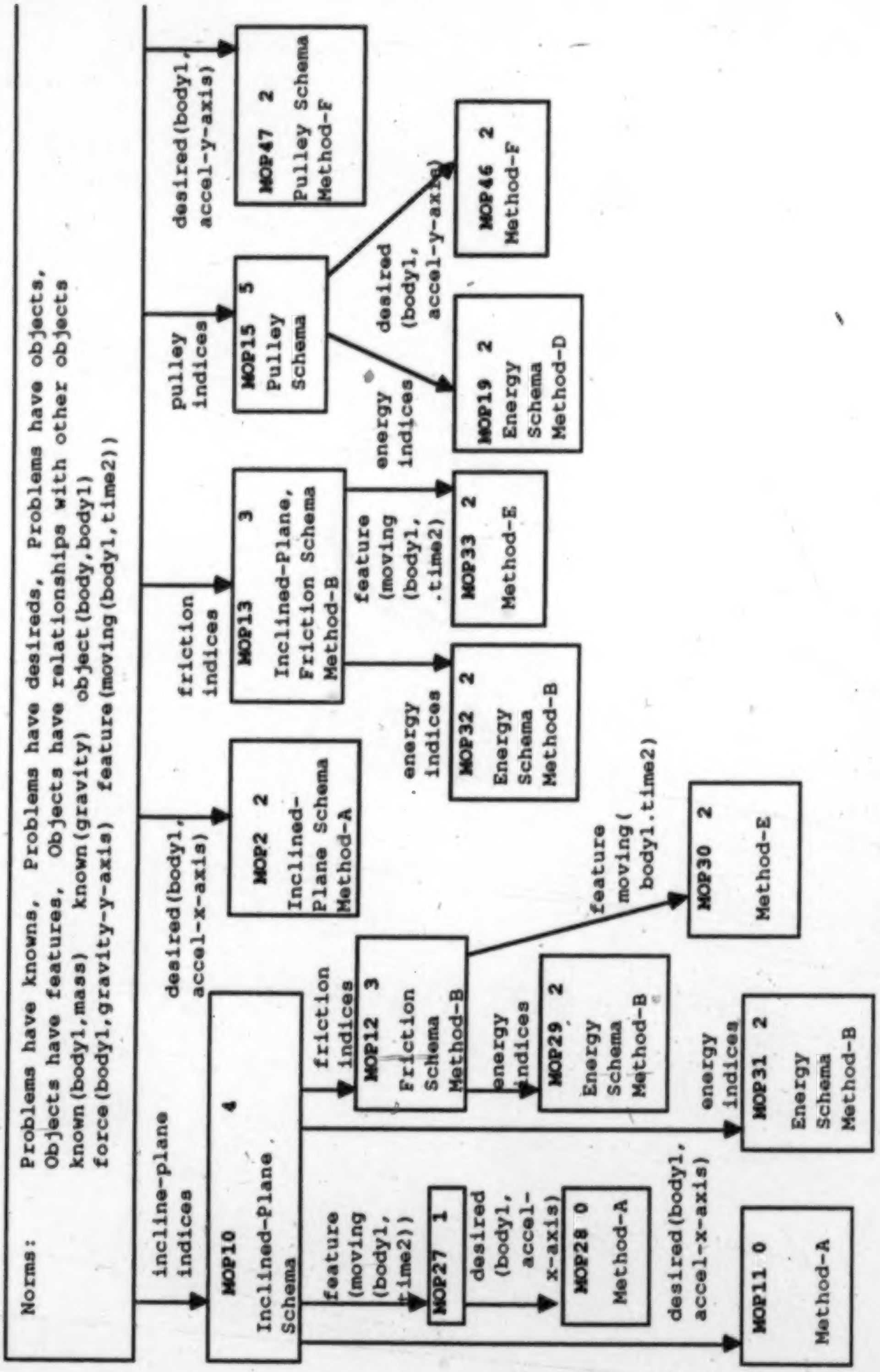

Norms:
Problems have knowns, Problems have desireds, Problems have objects,
Objects have features, Objects have relationships with other objects
known(body1,mass) known(gravity) object(body,body1)
force(body1,gravity-y-axis) feature(moving(body1,time2))
incline-plane indices
desired(body1, accel-x-axis)
friction indices
pulley indices
desired(body1, accel-y-axis)
MOP10 4
Inclined-Plane Schema
MOP2 2
Inclined-Plane Schema Method-A
MOP13 3
Inclined-Plane, Friction Schema Method-B
MOP15 5
Pulley Schema
MOP47 2
Pulley Schema Method-F
feature (moving (body1, time2))
MOP27 1
desired (body1, accel-x-axis)
MOP28 0
Method-A
desired(body1, accel-x-axis)
MOP11 0
Method-A
friction indices
MOP12 3
Friction Schema Method-B
energy indices
MOP29 2
Energy Schema Method-B
feature moving(body1.time2)
MOP30 2
Method-E
energy indices
MOP31 2
Energy Schema Method-B
energy indices
MOP32 2
Energy Schema Method-B
feature (moving (body1, .time2)
MOP33 2
Method-E
energy indices
MOP19 2
Energy Schema Method-D
desired (body1, accel-y-axis)
MOP46 2
Method-F

Figure 5.9a P-MOP network after Problem9 is indexed

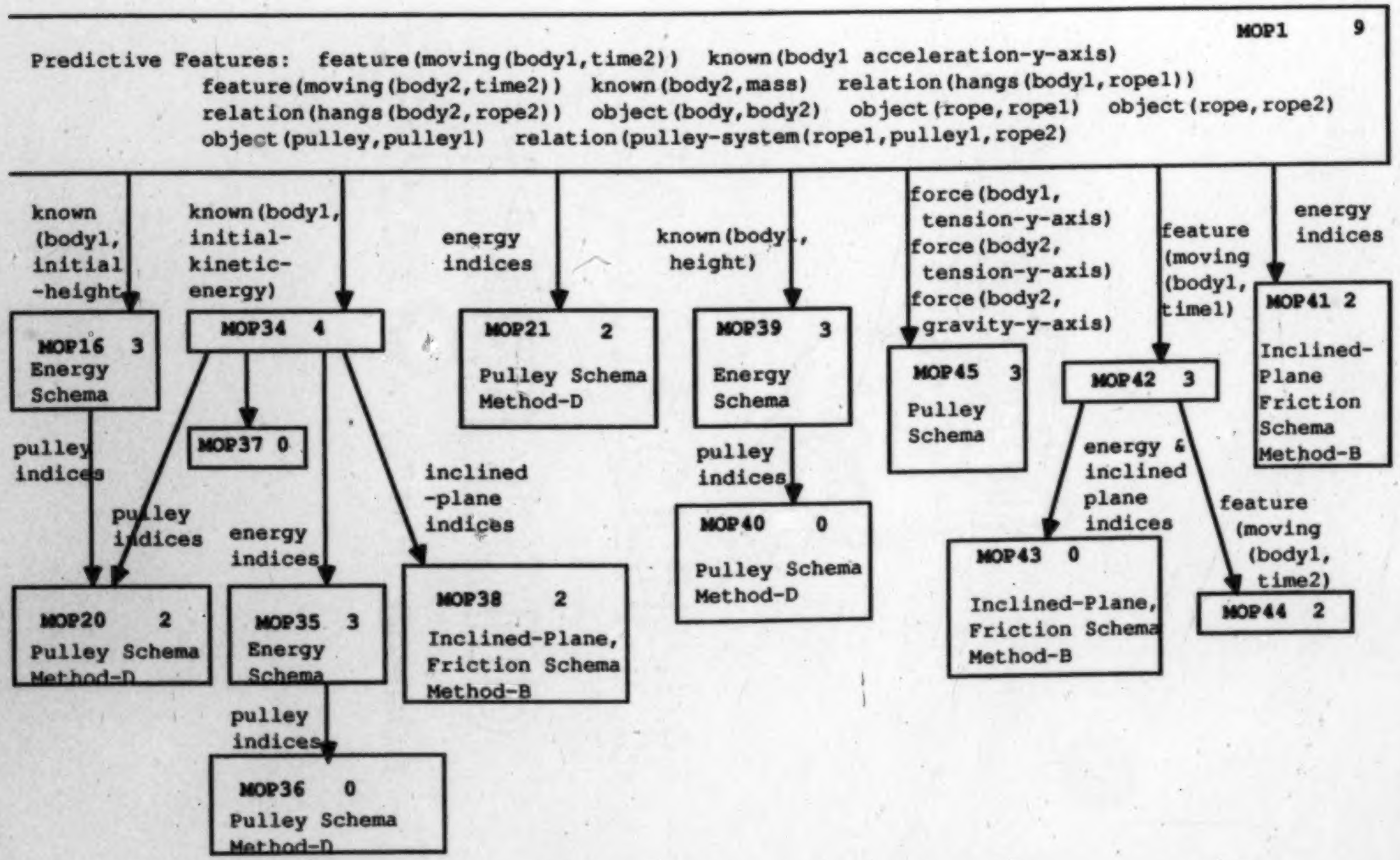

MOP1 9
Predictive Features: feature(moving(body1,time2)) known(body1 acceleration-y-axis)
feature(moving(body2,time2)) known(body2,mass) relation(hangs(body1,rope1))
relation(hangs(body2,rope2)) object(body,body2) object(rope,rope1) object(rope,rope2)
object(pulley,pulley1) relation(pulley-system(rope1,pulley1,rope2)
known (body1, initial -height)
known(body1, initial-kinetic-energy)
energy indices
known(body1, height)
force(body1, tension-y-axis) force(body2, tension-y-axis) force(body2, gravity-y-axis)
feature (moving (body1, time1)
energy indices
MOP16 3 Energy Schema
MOP34 4
MOP21 2 Pulley Schema Method-D
MOP39 3 Energy Schema
MOP45 3 Pulley Schema
MOP42 3
MOP41 2 Inclined-Plane Friction Schema Method-B
pulley indices
MOP37 0
pulley indices
energy indices
inclined -plane indices
pulley indices
energy & inclined plane indices
feature (moving (body1, time2)
MOP20 2 Pulley Schema Method-D
MOP35 3 Energy Schema
MOP38 2 Inclined-Plane, Friction Schema Method-B
MOP40 0 Pulley Schema Method-D
MOP43 0 Inclined-Plane, Friction Schema Method-B
MOP44 2
pulley indices
MOP36 0 Pulley Schema Method-D

Figure 5.9b P-MOP network after Problem9 is indexed

Problem 10:

A body of mass 5 kgs. starts moving from the top of an inclined plane of length 10 meters. The plane is inclined 30 degrees from the horizontal and the coefficient of friction is 0.05. What is the kinetic-energy of the body when it reaches the bottom of the plane?

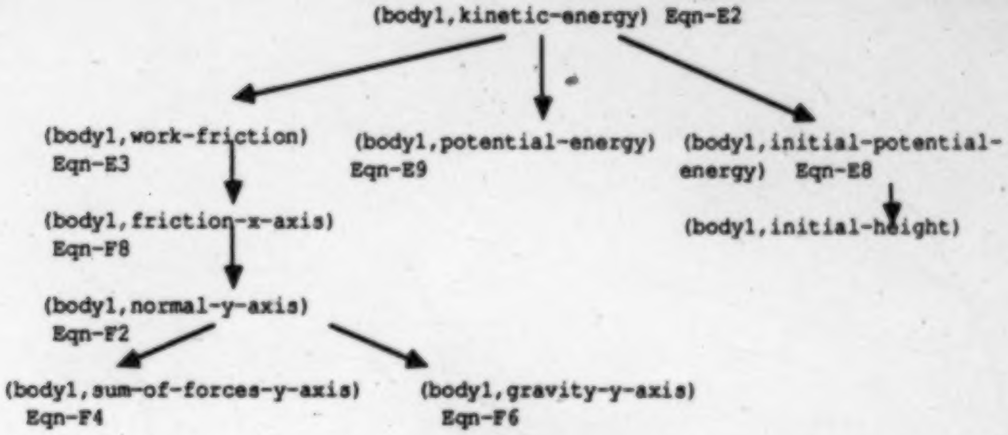
Solution:

EUREKA is able to pick up all relevant inferences from the sub-MOPs and problems indexed under MOP10 prior to using means-ends analysis to solve the problem.

P-MOP Reorganization:

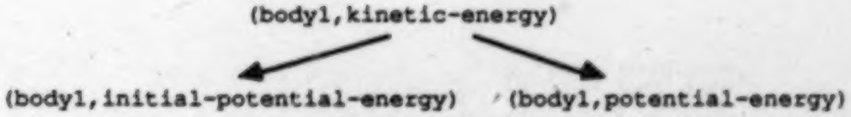
MOP34, of Figure 5.9b, becomes non-predictive because it indexes a number of problems but does not contain co-occurring norms. Non-predictive MOPs and their sub-MOPs disappear because non-predictive features are eliminated as indices into the P-MOP network. This is important because the P-MOP network is converging on a stable structure for this set of problems. Appropriate organizations of knowledge have their features, inferences and methods confirmed and will index more and more of the systems problem-solving experiences. In contrast, inappropriate organizations of knowledge have their features, inferences, and solution methods disconfirmed and moved lower in the network and MOP's containing only one norm turn non-predictive and are functionally forgotten.

The Energy Schema, MOP39, now has an inclined-plane sub-MOP (MOP51) and a pulley sub-MOP (MOP-40). Notice that the index to MOP39 is a known quantity. This organization supports the working-forward knowledge-development strategy of experts. The inclined-plane sub-MOP contains a new solution method, Method-G, that details an energy solution for inclined planes.


(body1,kinetic-energy) Eqn-E2
(body1,work-friction) Eqn-E3
(body1,potential-energy) Eqn-E9
(body1,initial-potential-energy) Eqn-E8
(body1,friction-x-axis) Eqn-F8
(body1,initial-height)
(body1,normal-y-axis) Eqn-F2
(body1,sum-of-forces-y-axis) Eqn-F4
(body1,gravity-y-axis) Eqn-F6

Method-G (Energy)

The Energy Schema (MOP39) also has a sub-MOP (MOP52) containing a new solution method. Method-H details the common steps between an inclined plane problem solved by an energy method and a pulley problem solved by an energy method.



Method-H (Energy)

A new inclined-plane MOP is indexed under MOP1 by the forces that occur in the problem.

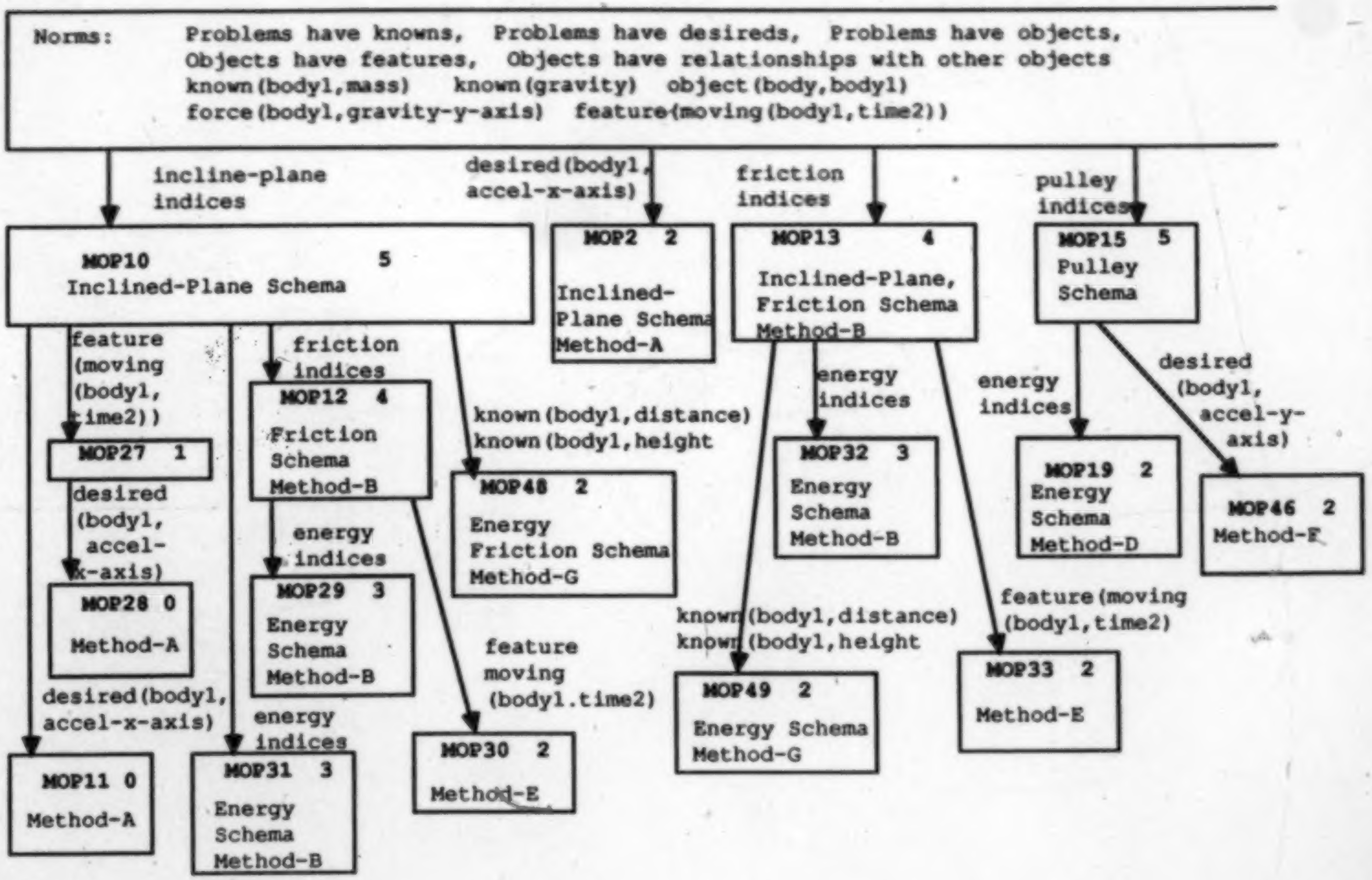


Figure 5.10a P-MOP network after Problem10 is indexed

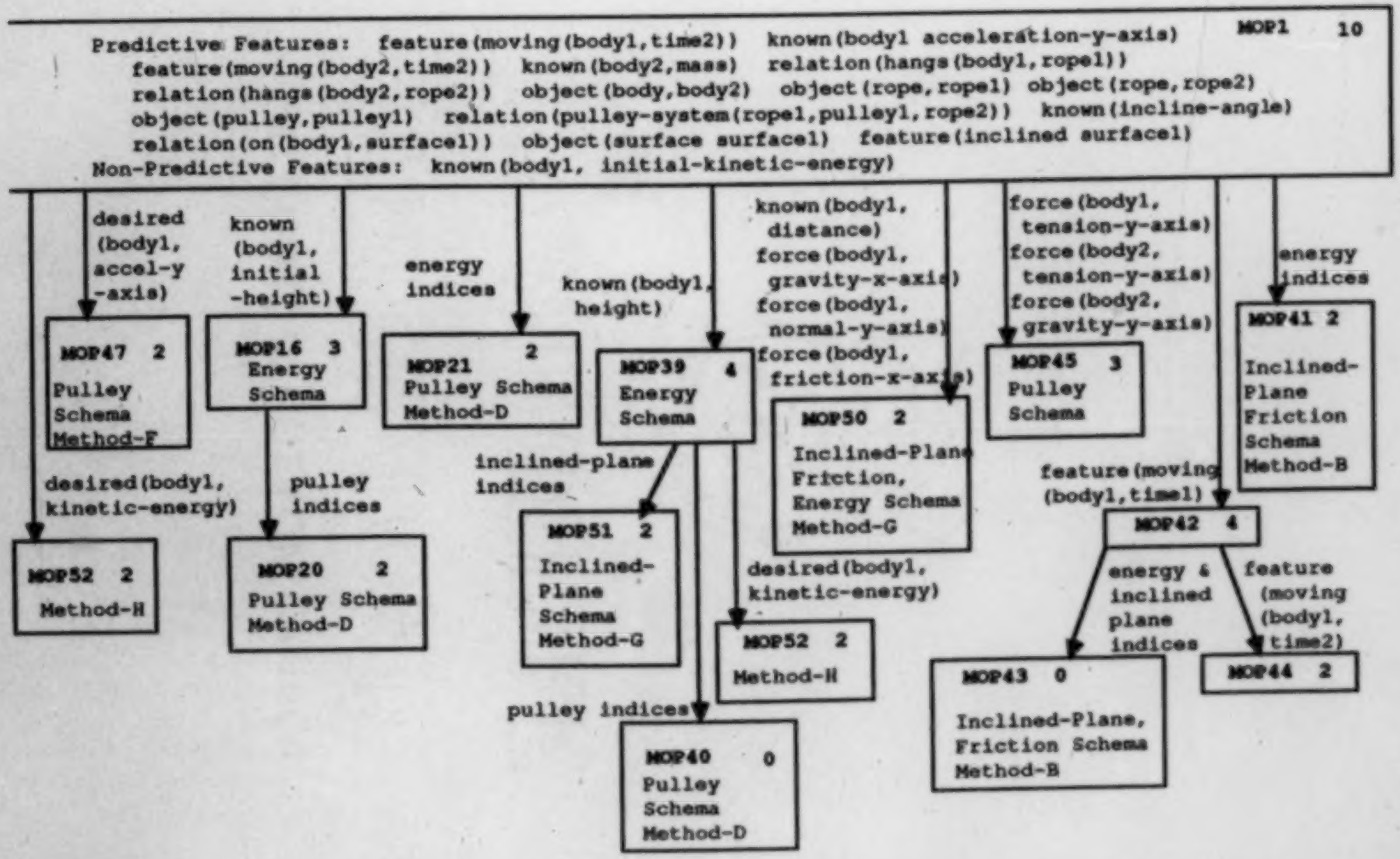

MOP1 10
Predictive Features: feature(moving(body1,time2)) known(body1 acceleration-y-axis)
feature(moving(body2,time2)) known(body2,mass) relation(hangs(body1,rope1))
relation(hangs(body2,rope2)) object(body,body2) object(rope,rope1) object(rope,rope2)
object(pulley,pulley1) relation(pulley-system(rope1,pulley1,rope2)) known(incline-angle)
relation(on(body1,surface1)) object(surface surface1) feature(inclined surface1)
Non-Predictive Features: known(body1, initial-kinetic-energy)
desired (body1, accel-y -axis)
MOP47 2
Pulley Schema Method-F
desired(body1, kinetic-energy)
MOP52 2
Method-H
known (body1, initial -height)
MOP16 3
Energy Schema
pulley indices
MOP20 2
Pulley Schema Method-D
energy indices
MOP21 2
Pulley Schema Method-D
known(body1, height)
MOP39 4
Energy Schema
inclined-plane indices
MOP51 2
Inclined-Plane Schema Method-G
pulley indices
MOP40 0
Pulley Schema Method-D
desired(body1, kinetic-energy)
MOP52 2
Method-H
known(body1, distance)
force(body1, gravity-x-axis)
force(body1, normal-y-axis)
force(body1, friction-x-axis)
MOP50 2
Inclined-Plane Friction, Energy Schema Method-G
force(body1, tension-y-axis)
force(body2, tension-y-axis)
force(body2, gravity-y-axis)
MOP45 3
Pulley Schema
feature(moving (body1,time1)
MOP42 4
energy & inclined plane indices
MOP43 0
Inclined-Plane, Friction Schema Method-B
feature (moving (body1, time2)
MOP44 2
energy indices
MOP41 2
Inclined-Plane Friction Schema Method-B

Figure 5.10b P-MOP network after Problem10 is indexed

Summary

In the final P-MOP network shown in Figure 5.10a and Figure 5.10b, many of the MOPs under the root-MOP (MOP1) are indexed by knowns or deeper features, such as forces, implied by features in physics problems. For example, the energy indices that lead to MOP21 are known = (body2, initial-kinetic-energy), known = (body2, height), known = (body1,initial-velocity-along-y-axis), and known = (body2,initial-velocity-along-y-axis). MOP50 is indexed by a known quantity and implied deep features (i.e., forces that occur in physics problems). The development of these indices at a high level in the P-MOP network supports the shift towards the forward-working strategy that characterizes expert problem-solving. However, another part of the P-MOP network (i.e., MOP10, MOP15) is still indexed with surface features and will continue to be preferred for traversal because they index a great number of problems. This is just a result of the nature of the particular problems EUREKA has solved. These problems were either inclined-plane problems or pulley problems so these surface features will appear in approximately half of the problems indexed. As EUREKA solves more problems, the frequency count of the known and deep features will also increase over time. If EUREKA were given a greater variety of surface features in problems solved with energy and force principles the number of problems indexed by any particular surface feature would be proportionally less than the number of problems indexed by knowns and deeper features of physics problems.

In order to better understand the knowledge structure that EUREKA has developed, I have abstracted and summarized the P-MOP network of Figures 5.10a and 5.10b. The resulting P-MOP network is shown in Figure 5.11. MOPs that contained only one feature and no solution methods are eliminated because they will eventually turn non-predictive and disappear. MOPs that will collapse over time (given more force and energy, inclined plane, and pulley physics problems) are also eliminated. For example, MOP12 will collapse into MOP10 because most inclined plane problems have a friction component. Finally, the sub-MOPs directly under MOP1 that index only a few problem experiences are not shown in

Figure 5.11. These MOPs exist, but EUREKA prefers to traverse MOPs that organize more problem-solving experiences. Notice that a number of these low frequency MOPs are indexed by "desired". Traversing high frequency MOPs allows EUREKA to move away from focusing on the desired quantity of a problem in a means-ends fashion. EUREKA prefers indices about knowns and the deeper features of the problem.

In the P-MOP network of Figure 5.11, sub-structures can be interpreted as surface feature schema (H - a pulley schema) with sub-schema detailing physics principles (I - a force sub-schema and J - an energy sub-schema). The P-MOP network also has a sub-structure that can be interpreted as an energy principle schema (K) with surface feature sub-schema (L - an inclined-plane sub-schema and M - a pulley sub-schema). These types of structures would be useful in situations that mention, for example, known energy quantities before detailing the surface configuration of the problem. These characteristics are similar to the interpretation of an expert's schema as shown in Figure 5.12 [Chi, Feltovich & Glaser, 1981]. If we focus on MOPs A, B, and C in Figure 5.11 the similarities to Chi et al.'s experts representation become even clearer. The expert's inclined-plane feature, in Figure 5.12, leads to Newton's force laws or conservation of energy principles. Newton's force laws in turn lead conditions of application and solution procedures. Similar structures appear in EUREKA's knowledge organization. In Figure 5.11, inclined plane indices lead to an Inclined-Plane-Friction schema (A). This schema in turn leads to either a force sub-schema (B) containing a force method or an energy sub-schema (C) containing an energy method. The sub-schema and its associated solution method that is used depends on the indices that are traversed from MOP-A. These indices can be viewed as conditions of applicabilty for the MOP (and associated solution method) that the index leads to.

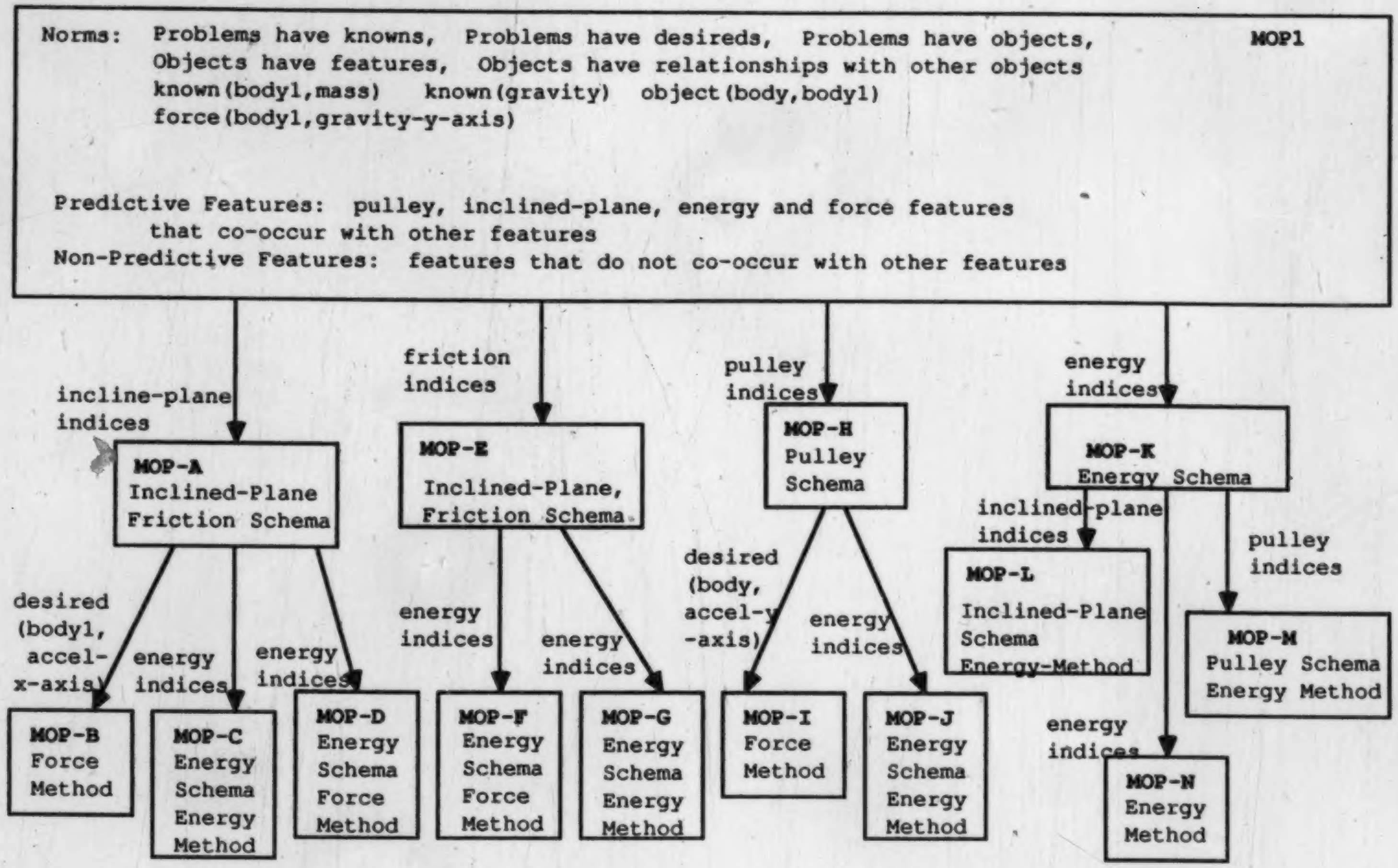

MOP1
Norms: Problems have knowns, Problems have desireds, Problems have objects,
Objects have features, Objects have relationships with other objects
known(body1,mass) known(gravity) object(body,body1)
force(body1,gravity-y-axis)
Predictive Features: pulley, inclined-plane, energy and force features that co-occur with other features
Non-Predictive Features: features that do not co-occur with other features
incline-plane indices
friction indices
pulley indices
energy indices
MOP-A Inclined-Plane Friction Schema
MOP-E Inclined-Plane, Friction Schema
MOP-H Pulley Schema
MOP-K Energy Schema
desired (body1, accel-x-axis)
energy indices
energy indices
energy indices
energy indices
desired (body, accel-y -axis)
energy indices
inclined-plane indices
pulley indices
energy indices
MOP-B Force Method
MOP-C Energy Schema Energy Method
MOP-D Energy Schema Force Method
MOP-F Energy Schema Force Method
MOP-G Energy Schema Energy Method
MOP-I Force Method
MOP-J Energy Schema Energy Method
MOP-L Inclined-Plane Schema Energy-Method
MOP-M Pulley Schema Energy Method
MOP-N Energy Method

Figure 5.11 P-MOP network

This Figure has been removed because of the unavailability of copyright permission.

Figure 5.12 Network representation of an experts schema of an inclined plane.
(From Chi et al., 1981, P137)

The P-MOP network shown in Figure 5.11, does not have a schema that details force principles with surface feature sub-schema. This is because inclined plane problems and pulley problems do not have a common force[1]. If EUREKA were given a wider variety of problems, with different surface features and solvable by force and energy methods, a force schema indexed by common forces would develop (i.e., a body on a flat table connected by a rope to another body hanging off the table, would have forces in common with both inclined plane problems and pulley problems). The force schema that would develop would have surface feature sub-MOPs.

EUREKA's goal was to model two aspects of novice-expert behaviour. Novices and experts organize their knowledge qualitatively differently. Novices organize their knowledge around surface features whereas an expert's organization includes principles of the domain. The trace demonstrates that EUREKA, with experience, organizes its knowledge around physics principles. The second aspect of novice-expert behaviour involves their problem solving method. Novices work backwards from what is desired in the problem whereas experts work forward from information given in the problem statement. This type of behaviour is demonstrated by EUREKA's growing preference for indices that indicate what is known in the problem rather than what is desired. Implementation of a procedure that utilizes the solution methods found in the P-MOP network would allow EUREKA to, with experience, rely less on the use of means-ends analysis.

---

[1] Body1's gravity force along the y-axis shows up as a norm on MOP1 and because of the choice of the reference frame for pulley problems, the y-axis in the direction of the bodies hanging from the pulley, body1 in a pulley problem does not have a x-axis gravity component.

Chapter 6

Discussion

The problem-solving trace demonstrated how EUREKA's final knowledge organization evolves from a novice-like organization based on surface features to an expert-like organization characterized by known quantities and abstract physics entities. In its novice stages, EUREKA has no option but to use means-ends analysis to solve problems. When a few problem solving experiences are indexed, the initial P-MOP network is organized around surface features. As more problems are solved, known quantities and inferred abstract physics entities become the discriminating and characteristic features on which organizational structure is based. EUREKA begins to prefer indices about known quantities and abstract physics entities like forces because, over time, they emerge as the "true" discriminating and characteristic features of problems. Intuitively, this result is completely reasonable: the "cover stories" used in physics problems are more variable than the underlying principles that characterize the appropriate inferences and solution methods. In this way, EUREKA's solution on later problems acquires a more forward-working, knowledge-development flavor because much of the knowledge needed to solve the problem is contained in the P-MOP network.

Although the previous chapter used the word "schema" in relation to particular P-MOPS, it is worth re-emphasizing that any single P-MOP does not correspond to a schema in the typical use of the term, i.e., as some complete cognitive unit that holds all the knowledge about some particular concept. Rather, the process of a single traverse along one particular path of nodes and indices is better viewed as one complete pattern match to a body of knowledge that is functionally analogous to a schema. Indices are traversed in an order determined by the number of problems they organize. In this respect, retrieval paths leading to more frequently occurring patterns are preferred. As noted above, the most frequently-occurring discriminating features are known quantities and inferred abstract entities.

An underlying assumption of this work is that knowledge restructuring underlies skill acquisition. EUREKA's restructuring mechanisms represent a different approach to the kind of knowledge compilation processes used by Anderson et al. [1981] in a production system framework. Functionally, EUREKA's mechanisms accomplish some of the same results. Unorganized declarative knowledge contained in the semantic network is used to construct goal-subgoal trees during problem solving. These goal-subgoal trees constitute a procedure that could be used on subsequent problems without repeating the retrieval of specific knowledge from semantic memory. However, there is a feature of these goal-subgoal trees that is worth underscoring. Because of EUREKA's "focus of attention" process, these goal-subgoal trees reflect the sequential order in which concepts were considered. Thus, the construction of goal-subgoal trees by the means-ends problem solver is really a temporal record of a "train of thought." Like the composition mechanism used in the ACT framework for modelling skill acquisition [Anderson, Kline, Greeno & Neves, 1981; Anderson & Neves, 1981], solution steps in EUREKA's goal-subgoal tree are associated with each other as a function of temporal ordering. However, unlike the large productions formed from composition, goal-subgoal trees are treated like any other pattern in the P-MOP network by EUREKA's restructuring mechanisms. Thus, they can be broken into sub-methods and become re-associated with different features of the evolving problem schemas. In this way, any spurious and incorrect temporal associations that associate particular goals and subgoals during one problem-solving episode are undone by these restructuring mechanisms - they end up on P-MOPS that are rarely or never traversed.

The episodic memory framework on which I based EUREKA is similar to Carbonell's [1983, 1986] work on analogical problem solving. In Carbonell's framework, past experiences are organized in a MOP network to enable the transfer of knowledge from relevant past experiences to the current problem. However, Carbonell

[1986] adapts solutions found in the network via a process called "derivational transformation". This process inspects the reasoning steps of the past solution in order to decide whether or not the solution is appropriate to the new situation. Reasoning steps include the subgoal structure of the problem, each alternative considered and rejected, reasons for the decisions taken, reasons for rejecting an alternative, dependencies among decisions, and pointers to knowledge that proved useful to the solution. In contrast, solutions found on P-MOPs do not carry as much information and are not open to this type of inspection. In the EUREKA framework, solution methods that reside on P-MOPs would be applied in an all-or-none manner during problem solving and are treated as patterns during knowledge restructuring. Any resemblance that EUREKA might have to analogical problem solving stems from the "reminding" aspect of the MOP framework [Schank, 1980]; it does not include any mechanisms for adapting previous solutions to new circumstances.

It is worth reiterating that Rumelhart and Norman's [1978] learning framework views analogical learning as a form of restructuring that modifies the organization of knowledge structures. In their framework, restructuring takes the form of *patterned generation* or *schema induction*. Patterned generation is the process by which a new schema is created by being patterned after the existing schema that best fits the current situation. The E-MOP reorganization mechanisms [Kolodner, 1984] are essentially forms of patterned generation. EUREKA's adaptation of these mechanisms creates new P-MOPs containing the norms, inferences and solution methods from the parent P-MOP that do not occur in the incoming problem representation. The newly-formed parent P-MOP does not contain information which was disconfirmed by the incoming problem representation. Thus, both the new parent P-MOP and the new sub-P-MOP are patterned after the old parent P-MOP. Similarly, the mechanisms which collapse a P-MOP into its parent P-MOP can be viewed as a form of schema induction. The trigger for this collapse is the recognition that the sub-MOP indexes a majority of its parent's events. In other words, this type of schema induction follows from noticing regularities and co-occurrences among existing schema.

The fact that EUREKA indexes the enhanced problem representation means that its patterned generation and schema induction is not limited to the problem's initial descriptors (e.g., the initial knowns, objects, features, relations, and desireds). Instead, abstract physics concepts and other inferences about object relations and features are eligible to become norms and indices. It is interesting to compare EUREKA's enhanced knowledge about objects, their features, and their relation to other objects with Novak's [1977,1980] notion of canonical physics objects. Novak's ISAAC system used schemas for defining and recognizing canonical physics objects such as "pivots" given a problem description. In order to direct recognition of a canonical object, these schemas must specify particular object features, the object's relation to other objects, and perhaps features and relations of those other objects as well. Clearly, the ability to recognize these abstract objects is an important kind of expertise as these kinds of canonical objects are important pieces of physics knowledge. In EUREKA, the kind of information that is grouped together on a P-MOP might support the automatic abstraction of canonical object schemas. For example, consider the the features, relations and inferences about particular object variables in the Pulley Schema (Mop 15) listed below:

```
Norms:        feature (moving (body1, time2))      feature(moving (body2, time2))
              relation (hangs (body1, rope1))      relation (hangs (body2, rope2))
              relation (pulley-system (rope1, pulley1, rope2)
              object (body, body2)                 object (rope, rope1)
              object (rope, rope2)                 object (pulley, pulley1)
              force (body2, gravity-y-axis)        force (body1, tension-y-axis)
              force (body2, tension-y-axis)        known (body2, mass)
              equal-to ((body1, acceleration-y-axis)(body2, acceleration-y-axis))
              equal-to ((body1, velocity-y-axis)(body2, velocity-y-axis))
              equal-to ((body1, tension-y-axis)(body2, tension-y-axis))
Inferences:   If     a body is hanging from a rope
              Then there is a force due to tension acting on the body.
              If     a body hangs from a rope
                     and another body hangs from another rope
                     and these two ropes and a pulley form a pulley-system.
              Then  the tension force acting on the two bodies are equal
                     and the acceleration of the two bodies are equal.
```

Although "pulley-system" is not the same flavor of canonical object as "pivot" or "lever," it serves to illustrate the following point. As a package, this knowledge seems to approximate what might correspond to the canonical definition of a "pulley-system." Although the problem descriptions used as input to EUREKA already included "pulley-system" as a relation, that simply reflects the level of granularity and sophistication of the semantic network knowledge-base. A more fine-grained representation of the objects, features, and relations involved with a pulley system could have been used in the initial problem description; the critical defining co-occurring object features and relations for "pulley system" would have grouped together on a P-MOP in the network. The problems I have tested EUREKA on do not involve objects that play roles such as pivots or levers. Nonetheless, it is an interesting possibility that functional definitions for them may emerge as a consequence of the P-MOP restructuring mechanisms. Unlike the ISAAC system, however, EUREKA's approximations to canonical object definitions are quite implicit. In addition, they do not drive the problem-solving process in the explicit and complex way that ISAAC used them. Once ISAAC recognized certain canonical objects, this knowledge was passed off to a geometric level which eventually solved the problems with the appropriate equations. For EUREKA, arriving at and instantiating this bundle of object features, relations, and inferences with the current problem representation simply serves to direct it to particular methodologies and equations.

Although full use of the stored P-MOP knowledge is not yet implemented in the current EUREKA system, the problem-solving trace makes it clear that EUREKA's restructured P-MOP knowledge supports the shift from a means-ends approach to a forward-working, knowledge-development approach. Larkin's [1981] ABLE system accomplishes a similar shift by developing very specific, proceduralized production rules. As noted earlier, this approach to skill acquisition in a production system framework is similar to EUREKA's framework when we view a traversal through the P-MOP network as a complex pattern match for accessing pieces of procedural knowledge, inferences, and equations that correspond to a complete schema. However, I believe that EUREKA's framework is more flexible then the production system formalism for capturing and organizing the qualitatively different types of knowledge - inferences, constraints, criteria and discriminating features, and procedural methods - which an expert has about different types of problems within a domain. The shift in problem solving strategy from means-ends to knowledge-development occurs *as a consequence of* the representation and restructuring of these different types of knowledge.

Kolodner's [1983] work on the SHRINK system also considered using episodic memory restructuring mechanisms as the means by which a system could become more expert-like on diagnosis problems. For that system, Kolodner proposed what she calls "diagnostic MOPs" to represent domain knowledge about categories of psychiatric disorders and "process MOPs" to represent strategic steps or stages in the diagnostic reasoning process. For example, SHRINK had a process MOP that was concerned with examining the patient data and choosing the predominant clinical features. As a step in the diagnostic reasoning process, this roughly corresponds to coming up with the problem representation, i.e., the correct collection and weighting of the relevant patient data. In describing a doctor's failure to correctly diagnose a patient's disorder, and the corrections he seemed to make to his own reasoning process, Kolodner proposes that the corresponding refinements to SHRINK would be in the process MOP concerned with the selection and weighting of predominant clinical features. This would impact diagnosis of similar cases in the future because these features are the ones that are used as indices into the diagnostic MOP network.

The important difference between SHRINK and EUREKA is that Kolodner proposes an explicit representation of strategic decisions in the form of process MOPs. EUREKA has no similar explicit representation of strategy. The goal for EUREKA was to account for a rather high-level strategy shift as a consequence of learning problem types that are characterized by known quantities and abstract entities like forces. Refining SHRINK's process MOPs to re-weight particular aspects of the patient's medical history seems to correspond to two aspects of EUREKA's design. First, recognizing that some features may be more important than those in the initial problem statement is indirectly covered by the indexing of the enhanced problem representation rather than just the initial problem description. As noted earlier, this serves to redefine the space of descriptors that are available as problem norms and indices because the entire problem-solving process is what EUREKA "remembers." Second, the refinements to a process MOP that are directions to re-weight certain features correspond to EUREKA's preference to traverse index paths that organize the most frequently-occurring patterns in the network. It takes EUREKA several problems to develop a "successful" problem-solving strategy, but EUREKA's goal was to move slowly from initially convoluted, albeit successful, solutions to ones that were also successful, but more expert-like, in their approach.

The advantages to representing strategic problem-solving knowledge in an explicit form that is distinct from the domain content knowledge have been discussed extensively in the knowledge-based systems literature [e.g., Clancey & Letsinger, 1981; Swartout, 1983]. The most obvious one is that it makes this knowledge open to learning and inspection mechanisms. As noted earlier, EUREKA achieves its particular strategy shift as a function of knowledge restructuring, and within that restructured network, the preferences for certain retrieval paths over others also implicitly corresponds to the emergence of certain strategies. Although the four stages proposed by Larkin [1985] seem to describe accurately what an expert does during problem-solving, there are no empirical findings to suggest that there are separate, explicit representations of these strategic stages. With the exception of drawing labelled diagrams, even the current EUREKA system develops its enhanced problem representation first by identifying relevant abstract physics features and entities such as forces and energies (Larkin's scientific stage). These abstract entities are associated with additional inferences and a solution methodology with appropriate equations (Larkin's mathematical stage). Again, these "stages" emerge naturally (although implicitly) as a consequence of knowledge restructuring in the network: the deep features such as the presence of particular forces and energies occur as norms or indices "high" in the network, and solution methodologies that re-state the problem in terms of the appropriate equations are typically found lower in the network. Thus, the order in which different types of concepts are added to EUREKA's enhanced problem representation in its expert stage is at least consistent with the stages proposed by Larkin [1985].

There are a number of design decisions that are worth considering in more detail, from the view point of psychological plausibility as well as general skill-acquisition systems. These design decisions are considered here in what I consider to be an order of "least" to "most" important.

First, EUREKA, like Kolodner's CYRUS system [1984], has a "perfect memory" for problems it has previously solved. In principle, this is not a strong objection against EUREKA as a general framework for simulating the novice to expert transition. An imperfect memory for past problems would simply add "noise" to the process and have its largest impact on how quickly the system evolves to a more expert-like organization.

Second, it may seem that EUREKA side-steps the difficult decision of recognizing when it has found all the relevant forces in using the two equations that sum forces along the x or y axis. Recall that when EUREKA believes it has solved either of these two equations, it asks the user to confirm that the equation is indeed solved. This is required because EUREKA does not know if it has found all the relevant forces. Certainly, recognizing that all the relevant forces have been identified is important, but some of that knowledge is precisely what EUREKA learns through the evolving problem schema: namely that certain problem features, object features, and object relations imply the presence of various forces. As EUREKA gains experience, its solutions are directed from the knowledge found on the P-MOPs and all the relevant forces are inferred prior to using these equations. I wanted to keep the initial EUREKA system as simple as possible by eliminating the complexities involved in failure-driven learning [Riesbeck, 1981]. For EUREKA, a failure would be failing to solve for the desired quantity. Instead, I assumed that a novice could struggle toward some solution, either alone or with the advice of a tutor, so that the entire learning process was completely success-driven.

A third assumption involves the control issue concerning what EUREKA does when it runs out of equations to use and therefore needs a new focus of attention. Recall that when EUREKA attempts to establish a new focus-of-attention, it first considers recently-inferred concepts, then relations between objects, then object features, and finally objects themselves. This list of leads serves as an implicit meta-strategy and may not be a correct characterization of novice behavior. This ordering has some important affects on the character of any particular solution but minimal effects, I believe, on what emerges over time after many problem-solving experiences. There are two reasons why this is so. First, as experience grows, solution methodologies are directed more from the P-MOP structures rather than the means-ends analysis problem solver. Therefore, this ordering plays a smaller and smaller role as experience continues. Second, if EUREKA constructed solutions by focusing on aspects of the problem in, say, a random fashion rather than the ordered list given above, then the solution methods would just be more convoluted and variable. Over time, this "noise" would wash out in this system as it would in most learning systems. It is important to understand more clearly how attention is directed when a novice problem-solver runs out of equations and more empirical data are needed. However, this issue does not impact how knowledge is restructured with experience, which is the important aspect of the EUREKA framework.

The fourth issue deals with the representation of the solution method. In the current implementation, there are no explicit associations between concepts in the goal-subgoal tree and the inferences that lead to the recognition or derivation of these concepts. The goal-subgoal tree is a temporally-ordered set of concepts considered during solution. The equation used to derive the concept is associated with it in the tree. During means-ends problem-solving, EUREKA choses a particular equation from a set of candidate equations that are related to the focus-of-attention. This choice is based on how many unknowns there are in each equation. This means that this decision is (implicitly) controlled by the currently known concepts as well as the currently derived concepts that stemmed from particular inferences and other equations. Presently, problem norms that appear on the condition side of inferences are linked to those inferences so that the inference "follows" the norms as they move around the network. Solution methods are handled quite independently: a method on a particular P-MOP is compared against the incoming method and common sub-methods remain on the P-MOP whereas idiosyncratic aspects of the methods are organized on sub-P-MOPS.

What is lacking in this current implementation is a direct association between each solution step and its "set of support." A solution step is a combination of a concept and an equation through which the concept can be derived. The set of support for a solution step would be the collection of known quantities, derived quantities, and inferences that were responsible for the selection of that particular equation from the other candidate equations for deriving that step's concept. In the present EUREKA system, all this knowledge does exist on a P-MOP, but it is not organized in this fashion. By matching the set of support for particular solution steps as it traverses the network, EUREKA would essentially be able to say "I know I can derive X," "I know I can derive Y," and so on during traversal without resorting to any eqvations. This kind of extension to the representation would be reflected in a new kind of quantity, "*can-be* derived" quantities, showing up in EUREKA's enhanced problem representation as it constructs a solution. Moving to this type of representation would underscore the knowledge-development strategy that EUREKA evolves.

There are two open issues involving EUREKA as a psychological process model. The first is a control issue involving the use of solution methodologies that reside in the P-

MOP network. The control between traversal of the P-MOP network using solution methodologies found on P-MOPs versus using means-ends analysis to solve for a certain quantity is unresolved. I view the traversal of a complete series of P-MOPs as a pattern match. Moreover, I view this match as an automatic process rather than a strategic decision. What happens after a complete pattern match is an open issue. For instance, should the entire P-MOP network be traversed before employing means-ends analysis or should means-ends analysis be eligible for use between pattern matches. Detailed studies of more levels of physics problem solvers (i.e., intermediate physics problem solvers) may be able to determine when a knowledge development strategy is used, when means-ends analysis is used, and what determines the switch of one strategy to another. In-depth studies of problem-solvers as they develop expertise in a domain is required.

The second open issue involves the control of the P-MOP restructuring mechanisms. The P-MOP network can be viewed as long-term memory and the parts of the P-MOP network traversed during problem-solving as working memory. In this view only the knowledge in working memory should be made available to the mechanisms. Therefore, only the parts of the P-MOP network activated by traversal would be available for restructuring. At this point, it is uncertain how this view would effect EUREKA's learning of problem prototypes.

I will now summarize the important contributions of this work. The EUREKA framework provides mechanisms that naturally and easily move along the continuum of expertise from a novice-like endpoint to a more expert-like endpoint. This framework not only accommodates both novice and expert problem-solving behavior but it seems particularly well-suited for organizing and restructuring the qualitatively different types of knowledge that seem to characterize experts' schemas in this domain. This is accomplished through the use of some very basic learning mechanisms such as generalization, discrimination, frequency, and patterned generation which are at the foundation of the MOP restructuring techniques. Most importantly, EUREKA's extensions to these techniques and their incorporation in a larger problem-solving framework have demonstrated that shifts in problem-solving strategies can emerge as a natural consequence of modelling the shifts in knowledge organization.

A second contribution of this work is that the EUREKA simulation implicitly models many of the empirical results and descriptive analyses of novice and expert problem-solving behavior. This includes the strategy shift reported by Simon and Simon [1978] and Larkin, McDermott, Simon, and Simon [1980], the stages of representation analysis provided by Larkin [1985], the knowledge organizational models reported by Chi, Feltovich, and Glaser [1981], and the evolution of canonical objects described by Novak [1977, 1980].

The third contribution lies in the empirical predictions that this framework suggests. The EUREKA framework suggests that certain orderings of problems may hasten the learning process. Investigation of EUREKA's sensitivity to ordering may provide insights relevant to similar educational issues. The EUREKA framework also suggests certain empirical predictions on "garden path" problems. A garden path problem would "set up" a certain problem type through extraneous information or a particular cover story, and then switch types by asking for an unexpected quantity. An analysis of an expert's protocol done by Chi et al. [1981] suggests that the features of the problem statement trigger inferences and schemas for the expert. Therefore, one might expect an expert to be led down the garden path because of these automatically generated inferences. The EUREKA model makes these predictions clear. A garden path problem would lead EUREKA, in its "expert stages", to certain areas of its P-MOP network, generating inferences and solution methodologies that subsequently prove irrelevant to the desired quantity.

Bibliography

Anderson, J.R. (1983). *The Architecture of Cognition,* Harvard University Press, Cambridge, Massachusetts.

Anderson, J.R. (1987). Skill aquisition: Compilation of weak-method problem solutions. *Psychological Review,* 94, 192-210.

Anderson, J.R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science,* 8, 87-129.

Anderson, J.R., Greeno, J.G., Kline, P.J. & Neves, D.M. (1981). Acquisition of problem-solving skill. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, J.J.: Lawrence Erlbaum Associates.

Buchanan, B.G., & Feigenbaum, E.A. (1978). DENDRAL and Meta-DENDRAL: Their applications dimension. *Journal of Artificial Intelligence* , 11, 5-24.

Carbonell, J.G., (1983). Learning by Analogy: Formulating and generalizing plans from past experience. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Vol.I. Tioga Publishing Company, Palo Alto, California.

Carbonell, J.G., (1986). Analogy in problem solving. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Vol.II. Morgan Kaufmann, Los Altos, Calif.

Chi, M.T.H., Feltovich, P.J., & Glaser, R. (1981). Categorization and representation of physics knowledge by experts and novices. *Cognitive Science,* 5, 121-152.

Chi, M.T.H., Glaser, R. & Rees, E. (1982). Expertise in problem solving. In R.J. Sternberg (Ed.), *Advances in the psychology of human intelligence,* Vol. 1, 7-76, Hillsdale, NJ: Erlbaum.

Clancey, W.J., & Letsinger, R., (1981). NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. *Proceedings of the 7th International Joint Conference on Artificial Intelligence,* 829-836.

Davis, R., Buchanan, B.G., & Shortliffe, E.H. (1977). Production rules as a representation for a knowledge-based consultation system. *Artificial Intelligence* , 8, 15-45.

de Groot, A. (1966). Perception and memory versus thought: Some old ideas and recent findings. In B. Kleinmuntz (Ed.), *Problem solving.* New York: Wiley.

Feigenbaum, E.A. & Simon, H.A. (1984). EPAM-like models of recognition and learning. *Cognitive Science,* 8, 305-336.

Fikes, R.E., Hart, P.E., & Nilsson, N.J. (1972) Learning and executing generalized robot plans. *Artificial Intelligence,* 3, 251-288.

Halliday, D., & Resnick, R. (1981). *Fundamentals of Physics.* New York: John Wiley & Sons.

Hayes, J.R., & Simon, H.A. (1976). The understanding process: Problem isomorphs. *Cognitive Psychology*, 8, 165-190.

Hayes-Roth F., Klahr P., & Mostow D. (1981). Advice taking and knowledge refinement: an iterative view of skill acquisition. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, J.J.: Lawrence Erlbaum Associates.

Hinsley, D.A., Hayes, J.R., & Simon, H.A. (1978). From words to equations: Meaning and representation in algebra word problems. In P.A. Carpenter & M.A. Just (Eds.), *Cognitive processes in comprehension*. Hillsdale, N.J.:Erlbaum.

Johnson, P.E., Duran, A.S., Hassebrock, F., Moller, J., Prietula, M., Feltovich, P.J., Swanson, D.B., (1981). Expertise and error in diagnostic reasoning. *Cognitive Science*, 5, 235-283.

Kolodner, J.L., (1983). Towards an understanding of the role of experience in the evolution from novice to expert. *International Journal of Man-Machine Studies*, 19, 497-518.

Kolodner, J.L., (1984). *Retrieval and organizational strategies in conceptual memory: A computer model*, Hillsdale, NJ: Erlbaum.

Kolodner, J.L., Simpson, R.L. & Sycara, K., (1985). A process model of case-based reasoning in problem solving. In *Proceedings of IJCAI-85*, Morgan Kaufmann, 284-290.

Kolodner, J.L., (1987). Extending problem solver capabilities through case-based inference. In *Proceedings of the Fourth International Workshop on Machine Learning*, Morgan Kaufmann, 167-178.

Laird, J., Rosenbloom, P., & Newell, A., (1986). *Universal sugoaling and chunking: The automatic generation and learning of goal hierarchies*, Kluwer Academic Publishers.

Langley, P. (1983). Learning search strategies through discrimination. *International Journal of Man-Machine Studies*, 18, 513-541.

Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, R. Neches (Eds.), *Production system models of learning and development*. The MIT Press Cambridge, Massachusetts.

Larkin, J.H. (1979). Processing information for effective problem solving. *Engineering Education*, 70, 285-288.

Larkin, J.H. (1981). Enriching formal knowledge: A model for learning to solve textbook physics problems. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, J.J.: Lawrence Erlbaum Associates.

Larkin, J.H. (1983). The role of problem representation in physics. In D. Gentner & A.L. Stevens (Eds.), *Mental Models*, Hillsdale, J.J.: Lawrence Erlbaum Associates.

Larkin, J.H. (1985). Understanding, problem representations, and skill in physics. In S.F. Chipman, J.W. Segal & R. Glaser (Eds.), *Thinking and Learning Skills: Research and Open Questions*, Vol.2., 141-159, Hillsdale, J.J.: Lawrence Erlbaum Associates.

Larkin, J.H., McDermott, J., Simon, D.P., & Simon, H.A. (1980). Models of competence in solving physics problems. *Cognitive Science*, 4, 317-345.

Lebowitz, M. (1983). Generalization from natural language text. *Cognitive Science*, 7, 1-40.

Lebowitz, M. (1986). Concept learning in a rich input domain: Generalization-based memory. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol.II. Morgan Kaufmann, Los Altos, Calif.

Lenet, D.B. (1976). AM: An Artificial Intelligence approach to discovery in mathematics as heuristic search. *Ph.D. dissertation*, Stanford University.

McDermott, J., & Larkin, J.H. (1978). Representing textbook physics problems. *Canadian society for computation studies of intelligence*. University of Toronto press.

McKeithen, K.B., Reitman, J.S., Rueter, H.H., and Hirtle, S.C., (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology*, 13, 307-325.

Michalski, R.S., (1983). A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol.I. Tioga Publishing Company, Palo Alto, California.

Minsky, M., (1975). A framework for representing knowledge. In P.H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.

Mitchell, T.M., Utgoff, P.E., & Banerji R., (1983). Learning by experimentation: Acquiring and refining problem-solving heuristics. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol.I. Tioga Publishing Company, Palo Alto, California.

Neches, R. (1987). Learning through incremental refinement of procedures. In D. Klahr, P. Langley, R. Neches (Eds.), *Production system models of learning and development*. The MIT Press Cambridge, Massachusetts.

Neves, D.M., & Anderson, J.R. (1981). Knowledge Compilation: Mechanisms for the automatization of cognitive skills., In J.R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, J.J.: Lawrence Erlbaum Associates.

Newell, A., & Simon, H.A. (1972). *Human problem solving*. Englewood Cliffs, N.J.: Prentice-Hall.

Newell, A., & Rosenbloom, P.S. (1981). Mechanisms of skill acquisition and the law of practice. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, J.J.: Lawrence Erlbaum Associates.

Novak, G. (1977). Representations of knowledge in a program for solving physics problems. *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, Mass.: MIT Press, 286-291.

Novak, G.S. Jr. & Araya, A., (1980). Research on expert problem solving in physics. *First National Conference jon Artificial Intelligence*, Stanford University, 178-180.

Pople, H.E. (1977). The formation of composite hypotheses in diagnostic problem solving: An exercise in synthetic reasoning. *Proceedings of the 5th International Joint Conference on Artificial Intelligence.* Cambridge, Mass.: MIT Press.

Reder, L. & Anderson J.R. (1980). A partial resolution of the pardox of interference: The role of integrating knowledge. *Cognitive Psychology* , 12, 447-72.

Riesbeck C.K., (1981). Failure-driven reminding for incremental learning. *Proceedings of the 7th International Joint Conference on Artificial Intelligence,* 115-120.

Riesbeck C.K., (1984). Knowledge reorganization and reasoning style. *International Journal of Man-Machine Studies,* 20, 45-61.

Rosenbloom, P.S. (1986). The chunking of goal hierarchies: A model of practice and stimulus-response compatibility. In J. Laird, P.Rosenbloom & A. Newell, *Universal subgoaling and chunking,* Kluwer Academic Publishers.

Rosenbloom, P.S., & Newell, A., (1986). The chunking of goal hierarchies: A generalized model of practice. In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Vol.II. Morgan Kaufmann, Los Altos, Calif.

Rumelhart, D.E. & Norman D.A. (1978). Accretion, tuning and restructuring: Three modes of learning. In J.W. Cotton & R.L. Klatzky (Eds.), *Semantic factors in cognition,* Lawrence Erlbaum Associates.

Samuel, A.L., (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development,* No. 3, 211-229.

Schank, R.C., (1977). *Scripts, Goals, Plans and Understanding,* Erlbaum Associates, Hillsdale, N.J.

Schank, R.C., (1980). Language and memory. *Cognitive Science,* 4, 243-284.

Schoenfeld, A.H., & Herrmann, D.J. (1982). Problem perception and knowledge structure in expert and novice mathematical problem solvers., *Journal of Experimental Psychology: Leaning, Memory, and Cognition,* 8, 484-494.

Simon, H.A., & Chase, W.G. (1973). Skill in Chess. *American Scientist,* 61, 394-403.

Simon, D.P., & Simon, H.A. (1978). Individual differences in solving physics problems. In R. Siegler (Ed.), *Childrens's thinking: What develops?,* Hillsdale, J.J.: Lawrence Erlbaum Associates.

Simon, H.A., (1983). Why should machines learn? In R.S. Michalski, J.G. Carbonell & T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach,* Vol.I. Tioga Publishing Company, Palo Alto, California.

Smith, E.E., Adams, N. & Schorr, D. (1978). Fact retrieval and the paradox of interference. *Cognitive Psychology* , 10, 438-64.

Soloway, E.M., & Riseman, E.M. (1977). Levels of pattern description in learning. *Fifth International Joint Conference on Artificial Intelligence,* Cambridge, Mass., 801-811.

Sussman, G.J., (1975). *A Computer Model of Skill Acquisition*. American Elsevier, New York.

Swartout, W.R., (1983). XPLAIN: A system for creating and explaining expert consulting programs. *Journal of Artificial Intelligence* , 21, 285-325.

Voss, J.F., Greene, T.R., Post, T.A. & Penner, B.C. (1983). Problem-solving skill in the social sciences. In G.H. Bower (Ed.), *The psychology of learning and motivation*., Vol. 17, Academic Press.

Winston, P.H., (1975). Learning structural descriptions from examples. In P.H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.